MASTER'S THESIS

Master of Science in Advanced Mathematics and Mathematical Engineering

Title: Fully post-quantum protocols for e-voting, coercion resistant cast as intended and mixing networks

Author: Ramiro Martínez Pinilla

Advisor: Paz Morillo Bosch

Department: Mathematics

Academic year: 2018

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Master in Advanced Mathematics and Mathematical Engineering

Master's thesis

Fully post-quantum protocols for e-voting, coercion resistant cast as intended and mixing networks

Ramiro Martínez Pinilla

Supervised by Paz Morillo Bosch

January, 2018

Thanks to my advisor Paz Morillo and the PhD student Nuria Costa for their help and guidance.

Abstract

In an electronic election several cryptographic proofs are implemented to guarantee that all the process has been fair. Many cryptographic primitives are based on the hardness of the discrete logarithm, factorization and other related problems. However, these problems are efficiently computable with a quantum computer, and new proofs are needed based on different assumptions not broken by quantum computers.

Lattice based cryptography seems one of the most promising post-quantum alternatives. In this thesis we present a coercion resistant cast as intended proof and a proof of a shuffle, both completely based on lattice problems as Inhomogeneous Short Integer Solution (ISIS) and Ring Learning With Errors (RLWE).

With the first we prove to the voter that his vote correctly encodes his voting option, without allowing him to prove to a third party that he has chosen a specific option, to avoid the possibility of vote selling.

Shuffles are permutations and re-encryptions of casted votes performed by mixing network nodes (mix-net nodes), so that the output can not be related with the input and nobody can link a decrypted vote with the voter who casted it. Given that the goal is to make the output not linkable to the input it is essential to provide a proof of it being a correct shuffle that has preserved the integrity of the votes, without deleting, adding of modifying any of them.

To prove both things we have constructed non interactive zero-knowledge proofs, from which anyone can be convinced that a statement is true (with overwhelming probability over a security parameter) without revealing any information about the elements that witness it being true.

Keywords

cryptographic protocols, e-voting, post-quantum cryptographic protocol, RLWE encryption, coercion resistant cast as intended, proof of a shuffle, mix-nets

Contents

1. Introduction

Electronic voting offers several advantages over traditional elections. It needs less infrastructure, it is less expensive, it obtains faster the final results, it is more flexible and it considerably simplifies the voting procedure for electors that are broad at the time of the election. In order to be considered an alternative, electronic voting needs to be as reliable as a traditional election. To do so it heavily relies on cryptography to guarantee that anonymity of the voters is preserved while at the same time ensuring that there is no fraud of any kind. Several cryptographic tools are used to achieve those properties.

Cryptography is a key component of our connected society. From personal communications to online shopping we want our data, either a personal message or a credit card number, to remain private. We also want to be sure that we are communicating with the person or entity we want, and not an impersonator. Those conditions are achieved by means of cryptographic constructions, whose security usually relies on the hardness of some well known computational problems.

Besides the theoretical computational complexity of the problems the actual computational power is taken into account, considering also predictable improvements on the future available computers. Quantum computers might substantially increase our computational capabilities, as they are based on a different computational paradigm where some problems considered hard on classical computers became efficiently solvable. This presents a problem for cryptography, as many security proofs rely on the hypothesis of the hardness of problems that are known to be quantum probabilistically polynomial time computable. Therefore, new cryptographic primitives have to be used, based on different hypothesis. That is, based on the assumed hardness of different problems that are believed to be not quantum efficiently solvable.

The goal of this work is to present new cryptographic protocols useful for electronic voting and to prove its security against a quantum capable adversary. We focus on proving that votes are casted as intended by the user and preserve its anonymity.

This thesis is structured in five sections. An introduction (section 1) in which the requirements for electronic voting are presented and how quantum resistance can be achieved. Section 2 contains the notation and definitions of all cryptographic primitives that are used, along with the family of mathematical problems that are considered. Section 3 is devoted to explain some of the protocols that guarantee that a given encrypted vote is valid and prove to the voter that it has been casted correctly. Finally in section 4 we present a mix-net node used to anonymize the encrypted votes. In the last section (5) we expose some conclusions and possible future work that could extend what has been presented here.

1.1 Electronic voting, privacy and verifiability

Electronic voting has the potential of improving our democracies providing new ways of participation. This key role implies that it has to be trustworthy, therefore properties ensuring that an election is fair have to be carefully defined and mathematically formalized.

Verifiable electronic voting has already been implemented in several countries [PCGK17], and it has to meet several requirements in order to be equivalent to a classical election. The Council of Europe has recently adopted new recommendations on standards for e-voting [CoE17]. Some of the usual general requirements are detailed in the following non exhaustive list:

- **Authenticity**: only eligible voters should be able to cast a vote.
- **Integrity**: the results must be obtained from the casted vote of the authorized voters, and no vote

can be modified, removed or added.

- **Anonymity**: it must not be possible to find a link between a voter and its choice.
- **Coercion resistance**: a voter should no be able to show to a third party how he voted.
- **Verifiability**: it should be possible to verify that the casted vote corresponds to the voter's choice, that his vote has been counted and there has been no irregularities during the process.
- **Accountability**: in case of any failure it should be possible to find the entity responsible for it.

Some of this requirements may seem contradictory. A voter should have access to a proof of his vote being casted as intended (verifiability), while at the same time he should not be able to prove it to someone else. This is required to avoid vote selling, as nobody will be willing to pay for a vote if there is no guarantee that the corrupted voter has chosen the option indicated by the corruptor (coercion resistance). one possible solution implies a proof of *either the vote corresponds to a particular option or this proof was generated by a particular voter* (in this context being generated by a particular voter means that it makes use of a secret only known by this particular voter). When the voter gets this proof from the voting device he knows that the device does not known his secret information, hence the vote really encodes his option. Although, if the voter shows this proof to any other it has no value, as it could have been generated just knowing the secret information. This would be handled by a cryptographic protocol explained in detail in section 3.

Votes are encrypted to hide its content (anonymity). At the same time, in order to guarantee that only eligible voters vote (authenticity) they digitally sign their encrypted votes. Those encrypted and signed votes are published in a *bulletin board*, a public place where information can be added by an authority but not modified. Those votes can not be decrypted directly (as they are linked to the author by the signature), and several methods exist to guarantee (verifiability) that the content of the votes is not modified (integrity) while making impossible to link the results to particular voters (anonymity). We propose a mix-net protocol (a collection of mixing nodes that sequentially re-encrypt and permute the votes) to handle this issue in section 4.

An electronic voting scheme has to specifically state what can or cannot be compromised in case some of the parties involved are malicious, and how that could be detected. For example, if the server that receives the votes is malicious it can omit one of the received votes, but then it would create a discrepancy with the bulletin board, and the voter would notice. If it modifies or creates new votes it would fail trying to forge their digital signatures, therefore it would also be noticeable by anyone verifying the election. One of the most critical points is the decryption process. If only one entity holds the decryption secret key it could decrypt all signed votes and break anonymity. To avoid this scenario threshold decryption is used, where the secret key is shared among n parties. Each of them can compute a partial decryption, but at least t of them are needed to recover the message (with $t \leq n$). Therefore a malicious coalition of $t' < t$ parties can not recover the message.

This observation would be important in section 4, where several nodes perform the same anonymizing operation on the votes. The protocol is designed in such a way that it does guarantee that a malicious node can not modify the content of the votes. However a malicious voter could not anonymize anything (for example, revealing its random coins), and this property is achieved only if at least one of the nodes is honest. This is the same approach followed by traditional elections, where all process are performed by several people, and our guarantees come from not all of them being malicious.

A complete election scheme that fulfills all the requirements consists of several servers and protocols to specify the whole voting process. The two protocols presented in this work are key pieces of any e-voting scheme.

1.2 Quantum computers

The idea of a quantum computer has been proposed for some decades. Even if current prototypes are very limited many companies and research centers are trying to build larger quantum computers, and the theoretic results tell as that they would have cumbersome implications for cryptography.

While classical computers store information in bits, a 0 or a 1, quantum computers take advantage of quantum physics and stores information in the so called *qubits*, a linear superposition of the base states of a mechanical system. Using the bra-ket notation $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$.

A quantum 2-qubit state could be $|\psi\rangle = \alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle$. One usual misunderstanding is to think that this implies that quantum computers are exponentially faster than classical computers, or are able to solve any NP problem in polynomial time using as input a linear superposition of all witnesses. This is not the case, as we also need to understand how the output is read. When we obtain $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ as an output we measure it and we get $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$. The naive approach for solving NP problems exposed before would only return a 1 with a probability equal to the proportion of valid witnesses, which is no advantage compared with a classical algorithm.

In fact no quantum algorithm is known able to solve an NP-hard problem in polynomial time. However there are several *NP* problems not known to be in *P* with classical computation that can be solved polynomially with a quantum computer. The most important ones from a cryptographic point of view are the discrete logarithm and the factorization problem, that can be solved using the Shor's algorithm [Sho99].

This implies that many cryptographic protocols, whose security hypothesis is related to one of this problems, would became insecure in presence of a quantum computer. In some cases this might be seen as a future problem. Using today a quantum insecure protocol to authenticate a user is perfectly acceptable, as no one has access to a powerful enough quantum computer, hence the user can not be cheating. We would only need to replace those protocols when quantum computers start to be builded. Although, using a quantum vulnerable protocol to ensure the privacy of a secret is a bad idea, as some malicious adversary may store this information until he has access to a quantum computer, and then obtain the secret. This is particularly important in case of sensitive information, as it is the case for the voting choice in an electronic voting, where privacy should be preserved long term, even several decades after the election was held.

There are two differentiated approaches to deal with this problem. Quantum cryptography bases its security on quantum physical properties of the communication signal. This requires special equipment and communication channels. While this could be implemented in very specific environments that require this level of security it may not be necessary in the general case.

While some hard problems became efficiently solvable with a quantum computer there are other problems suitable for cryptography for which there is no known quantum algorithm able to solve them efficiently. Sometimes the level of security decreases, that is, if a problem requires an exponential number of operations to break its security, quantum computer also requires an exponential number, but with a lower constant in the exponent. This implies that the security parameters have to be stablished taking quantum computers into account, and then they would still be secure. Many symmetric encryption, hash based, code based, isogeny based, multivariate polynomials and lattice based schemes are believed to be secure against a quantum capable adversary, and belong to the so called post-quantum cryptography.

Post-quantum cryptography uses classical problems and classical algorithms, hence it can be implemented in any regular computer or device, making it the most appealing alternative. We would devote our work to one specific family of post-quantum primitives, lattice based cryptography. Lattice based problems are the most promising family of post-quantum problems, as they allow to build from signatures to public key encryption schemes. The main drawback of most post-quantum alternatives is the size of the ciphertexts and keys compared with their classical counterpart, specially when considering them integrated into a chip, but they are usable on any regular computer or mobile device, as has been tested in an experiment driven by Google to use *newhope*, a lattice based key exchange mechanism [ADPS16], in its browser communications [Bra16].

1.3 Our contribution

Our contribution consists in two different proofs. First, in section section 3 we show how to implement a coercion resistance cast as intended proof as the one proposed in [GC16], using only existing lattice based cryptographic primitives. It is a direct application of those primitives.

The main contribution of this thesis is the proof of a shuffle for a lattice based mix-net presented in section 4. It is based on the existing proof of Bayer and Groth for ElGamal encryptions [BG12], but in this case it is not a direct adaptation. Several modifications have been made to take into account the technical particularities of lattice cryptography in the online part of the proof, while the second offline part has been completely redesigned. We think this is an important contribution, as it is the first proof of a shuffle fully based on lattice based cryptography. This implies that it not only provides long term privacy, but it also would be sound in a post-quantum scenario.

2. Preliminaries

In this section we fix the notation, give a background on the cryptographic primitives that are used in this thesis, explain how we can prove knowledge of something without revealing it with a zero-knowledge proof and finally introduce lattices, lattice related problems that allow us to build cryptography and the particular proposals that we use in this work.

2.1 Notation

We represent column vectors by boldface lower-case roman letters $\boldsymbol{u}$ or $\boldsymbol{v}$ and row vectors with their transpose $\boldsymbol{v}^\intercal$. Matrices are denoted by boldface upper-case roman letters, $\boldsymbol{M}$ or $\boldsymbol{A}$. Given two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{Z}_q^N$, we write the standard inner product in $\mathbb{Z}_q^N$ as $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_{i=1}^{N} u_i v_i$.

We write $y \leftarrow \mathcal{A}(x)$ when the deterministic algorithm $\mathcal{A}$ outputs y on input x. If $\mathcal{A}$ is a probabilistic algorithm then we denote by $y \xleftarrow{\$} \mathcal{A}(x)$ the process of running $\mathcal{A}$ on input x with random coins and getting an output y. We also write $y \xleftarrow{\$} Y$ for sampling y uniformly at random from a set Y and $y \xleftarrow{\$} D$ for sampling y according to a probability distribution D.

Some elements from different cryptographic constructions play a similar role and are usually denoted in the literature with the same letter. We use a subindex to distinguish those variables, a_{E} would be an element related to encryption while a_{C} would be an element related to a commitment.

Another important notation issue arises from the use of secret permutations. Given a general vector $(v_1, v_2, \dots, v_n)$ and its permutation $\left(v_{\pi^{-1}(1)}, v_{\pi^{-1}(2)}, \dots, v_{\pi^{-1}(n)}\right) = (v'_1, v'_2, \dots, v'_n)$ using a permutation π they may be indexed in different ways. Sometimes $v_{\pi(i)}$ is indexed with i, that is, we know this element is the one with index the image of i by the permutation, but as the permutation is secret the actual value $j = \pi(i)$ is unknown. Some other times $v_{\pi(i)}$ is indexed by $j = \pi(i)$, that is, we write $\pi(i)$ for the reader to know where this element came from but when published is only labeled as v'_j, preserving the secret of $i = \pi^{-1}(j)$. This would be particularly explained in each case where there may be doubts.

Definition 2.1 (Negligible function)**.** A positive definite function $f(n)$ is said to be *negligible* if it decreases faster than the inverse of every positive polynomial. That is:

$$\forall c > 0 \quad \exists n_0 \in \mathbb{N} \quad | \quad \forall n \geq n_0 \quad f(n) < \frac{1}{n^c}$$

In all the computational hypothesis we consider that a problem is hard if no Probabilistic Polynomial Time adversary is able to solve it with non negligible probability. We abbreviate this as a PPT adversary. If it is not explicitly stated otherwise we will consider quantum adversaries.

We denote the unary representation of λ as 1^λ. This notation is useful as the computational complexity of an algorithm is determined by the cost as a function of the size of the input. While the size of n is $\log n$ the size of 1^λ is λ, which is more convenient for directly expressing the bits of security of a protocol without the need of taking logarithms.

2.2 Cryptographic primitives

A robust electronic voting scheme consists on several protocols that make an extensive use of many cryptographic primitives. Here we present in detail the main ones, public encryption, commitments and trapdoor functions.

2.2.1 Public encryption

A public key encryption scheme is intended to be used as a way of ensuring privacy. A public and a private key are generated. Anybody can use the public key to encrypt a message, knowing the secret key is possible to recover (decrypt) the message, but without it no relevant information can be extracted efficiently.

This can be formalized in the following way:

Definition 2.2 (Public Key Encryption Scheme)**.** A *public key encryption scheme* consists on three algorithms:

- **Gen**: the generator algorithm takes a security parameter 1^λ and outputs a pair of keys, sk and pk, the secret and the public key respectively. It also defines the message space $\mathcal{M}_\lambda$ and the ciphertext space $\mathcal{C}_\lambda$. $(sk, pk) \xleftarrow{\$} \mathtt{Gen}\left(1^\lambda\right)$
- **Enc**: the encryption algorithm takes as input a message $m \in \mathcal{M}_\lambda$ and a public key pk and produces a ciphertext $c \in \mathcal{C}_\lambda$. $c \xleftarrow{\$} \mathtt{Enc}\left(m; pk\right)$
- **Dec**: the decryption algorithm takes as input a ciphertext c and a secret key sk and outputs a message m'. $m' \xleftarrow{\$} \mathtt{Dec}\left(c; sk\right)$

And verifies:

$$\Pr\left[m \neq m' \middle| (sk, pk) \xleftarrow{\$} \mathtt{Gen}\left(1^\lambda\right),\ c \xleftarrow{\$} \mathtt{Enc}\left(m; pk\right),\ m' \xleftarrow{\$} \mathtt{Dec}\left(c; sk\right)\right] \in \mathrm{negl}(\lambda)$$

This definition ensures that anyone can encrypt messages, as the algorithms and the public key are known. People knowing the secret key can recover, with overwhelming probability, the secret message that is encrypted in the ciphertext. The first intuition about a public key encryption scheme tells us that besides this we also need that the original message should not be efficiently obtainable without knowing the secret key. This can be formalized with the following security definition:

Definition 2.3 (Public Key Encryption One Way Chosen Plaintext Attack Secure (PKE-OW-CPA Secure))**.** A public key encryption scheme is said to be *OW-CPA secure* if for all PPT adversaries $\mathcal{A}$ the following holds.

$$\Pr\left[m' = m_* \middle| (sk, pk) \xleftarrow{\$} \mathtt{Gen}\left(1^\lambda\right),\ m_* \xleftarrow{\$} \mathcal{M}_\lambda,\ c_* \xleftarrow{\$} \mathtt{Enc}\left(m_*; pk\right),\ m' \xleftarrow{\$} \mathcal{A}\left(c_*, pk, 1^\lambda\right)\right] \in \mathrm{negl}(\lambda)$$

However, usually this is not enough. An encryption scheme that only modifies the second half of the message may be PKE-OW-CPA secure if it is hard enough to recover this second half, but reveals a lot of information about the message. In our electronic voting process we use the following standard stronger security definition:

Definition 2.4 (Public Key Encryption Indistinguishable Chosen Plaintext Attack Secure (PKE-IND-CPA Secure))**.** A public key encryption scheme is said to be *IND-CPA secure* if for all PPT adversaries $(\mathcal{A}_1, \mathcal{A}_2)$ the following holds.

$$\left| \Pr \left[b' = b \;\middle|\; \begin{array}{c} (sk, pk) \xleftarrow{\$} \mathtt{Gen}\left(1^\lambda\right),\ (m_0, m_1, aux) \xleftarrow{\$} \mathcal{A}_1\left(pk, 1^\lambda\right),\ b \xleftarrow{\$} \{0,1\}, \\ c \xleftarrow{\$} \mathtt{Enc}\left(m_b; pk\right),\ b' \xleftarrow{\$} \mathcal{A}_2\left(c, pk, aux, 1^\lambda\right) \end{array} \right] - \frac{1}{2} \right| \in \mathrm{negl}(\lambda)$$

That is, the encryptions of two different messages should not be distinguishable (the advantage over a random guess has to be negligible), even if we allow the adversary to choose those messages. Note that this definition implies that the algorithm has to be probabilistic.

We prove security of an encryption scheme under certain hypothesis proving the existence of a reduction that implies that a PPT algorithm breaking the security of the encryption could be used to disprove the hypothesis. We are interested in public key encryption schemes whose security is based in hypothesis believed to be true in a quantum scenario.

2.2.2 Commitments

Sometimes we want to hide a message not to communicate with someone but to prove that it was fixed some time before it is revealed and has not been modified. This is the idea of a commitment scheme. Given a message m we build a commitment c and an opening d and publish the first. Everyone knows the commitment, but not the message underneath. At some point in the future if we publish the message and the opening everybody should be convinced that the commitment was generated from this particular message and no other, and then they know that the message we just revealed was predetermined before we published the commitment. This construction would be really useful to build an interactive zero-knowledge proof, where the order in which the information is exchanged is really relevant.

More formally:

Definition 2.5 (Commitment Scheme)**.** A *commitment scheme* consists on three algorithms:

- **Gen**: the generator algorithm takes a security parameter 1^λ and outputs a public key pk. $pk \xleftarrow{\$} \mathtt{Gen}\left(1^\lambda\right)$
- **Com**: the commitment algorithm takes as input a message m and a public key pk and produces a commitment c and an opening d. $(c, d) \xleftarrow{\$} \mathtt{Com}\left(m; pk\right)$
- **Ver**: the verification algorithm takes as input a commitment c, a message m, an opening d and a public key pk and accepts, 1, or rejects, 0. $\mathtt{Ver} : \{c, m, d; pk\} \to \{0, 1\}$

And verifies the following three properties:

- **Correctness**: if the commitment has been built correctly and the valid message and opening are published the verifier algorithm accepts with overwhelming probability:

$$\left(1 - \Pr\left[1 \leftarrow \mathtt{Ver}\left(c, m, d; pk\right) \middle| pk \xleftarrow{\$} \mathtt{Gen}\left(1^\lambda\right),\ (c, d) \xleftarrow{\$} \mathtt{Com}\left(m; pk\right)\right]\right) \in \mathrm{negl}(\lambda)$$

- **Binding**: a commitment can only be correctly opened to one message. This property can be perfect or computational.

  A commitment scheme is perfectly binding if:

$$1 \leftarrow \mathtt{Ver}\left(c, m, d; pk\right) \wedge 1 \leftarrow \mathtt{Ver}\left(c, m', d'; pk\right) \implies m = m'$$

  A commitment scheme is computationally binding if, for all PPT adversaries $\mathcal{A}$:

$$\Pr\left[\begin{array}{c}\left(1 \leftarrow \mathtt{Ver}\left(c, m, d; pk\right)\right) \\ \wedge\left(1 \leftarrow \mathtt{Ver}\left(c, m', d'; pk\right)\right) \\ \wedge m \neq m'\end{array} \middle| pk \xleftarrow{\$} \mathtt{Gen}\left(1^{\lambda}\right),\ (c, m, m', d, d') \xleftarrow{\$} \mathcal{A}\left(pk, 1^{\lambda}\right)\right] \in \mathrm{negl}(\lambda)$$

- **Hiding**: from a well constructed commitment c it should not be possible to recover the message m. Again, depending on how we formally describe this we obtain different definitions.

  A commitment scheme is perfectly hiding if a commitment can be opened to any message:

$$\text{given}\ \ (c, d) \xleftarrow{\$} \mathtt{Com}\left(m; pk\right), \forall m'\ \exists d' \quad | \quad 1 \leftarrow \mathtt{Ver}\left(c, m', d'; pk\right)$$

  A commitment scheme is computationally hiding if for any PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$:

$$\left|\Pr\left[b = b' \middle| \begin{array}{c} pk \xleftarrow{\$} \mathrm{Gen}(1^{\lambda}), \quad (m_0, m_1, aux) \xleftarrow{\$} \mathcal{A}_1(pk) \\ b \xleftarrow{\$} \{0,1\}, \quad (c, d) \xleftarrow{\$} \mathrm{Com}(m_b; pk), \quad b' \xleftarrow{\$} \mathcal{A}_2(c, aux) \end{array}\right] - \frac{1}{2}\right| \in negl(\lambda)$$

From this definition it is easy to see that one commitment scheme can not be at the same time perfectly hiding and perfectly binding. In section 2.4.4 we show the commitment scheme we use in our protocols, which is perfectly hiding (with overwhelming probability) and computationally binding under the RLWE assumption (that is explained in section 2.4.2). This will be crucial for us, as we will not reveal directly the openings of any commitment, but instead we will reveal the opening of a polynomial relation between the messages inside the commitments, therefore we need long term privacy for the messages of the initial commitments with a long term post-quantum hiding property.

2.2.3 Trapdoor functions

Definition 2.6 (Trapdor Function)**.** A *trapdoor function* is a function $f : X \to Y$ such that given $x \in X$ we have that $f(x) = y$ is efficiently computable, but given $y \in Y$ it is hard to obtain a preimage $x' \in f^{-1}(y)$ unless some secret information sk_f is known.

- **Efficiency**: Given a security parameter 1^{λ} a generator efficiently samples a function f with a trapdoor sk_f from the family of trapdoor functions. Computing $f(x)$ is efficient for any x in X.
- **Trapdoor**: There exists a PPT algorithm that given $f(x)$ and sk_f efficiently computes $x' \in X$ such that $f(x') = f(x)$.
- **Security**: For any PPT algorithm $\mathcal{A}$:

$$\Pr\left[f(x') = f(x) \middle| (f, sk_f) \xleftarrow{\$} \mathrm{Gen}(1^{\lambda}), x \xleftarrow{\$} X, x' \xleftarrow{\$} \mathcal{A}(f, f(x))\right] \in negl(\lambda)$$

The hardness of the preimage computation should be based on some computational hypothesis.

2.3 Zero-knowledge proofs

As we have previously explained an electronic voting scheme has to be verifiable while preserving privacy, and zero-knowledge proofs play a central role. If we want to proof that some element is a valid ciphertext of a voting option we can prove that we know some random coins such that the element is a valid ciphertext of that voting option with those random coins, without revealing any information of them. We proof some proposition proving knowledge of a witness of this proposition being true.

One particular example of a zero-knowledge proof of knowledge are Σ-protocols. Let R be a binary relation, that is, R is a subset of $\{0,1\}^* \times \{0,1\}^*$ with the following restriction, if $(x, w) \in R$ then the size $|w|$ of w is at least $p(|x|)$ for some polynomial p. For $(x, w) \in R$ we consider x as an instance of a particular problem and w a solution for this instance, usually called witness.

Definition 2.7 (Σ-protocol)**.** A Σ-protocol is a protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ in which, given an x, $\mathcal{P}$ tries to convince $\mathcal{V}$ that it knowns a witness w such that $(x, w) \in R$. We use the following notation:

$$\text{ZK-proof}\left[w \,\middle|\, (x, w) \in R \right]$$

A Σ-protocol consists of three movements:

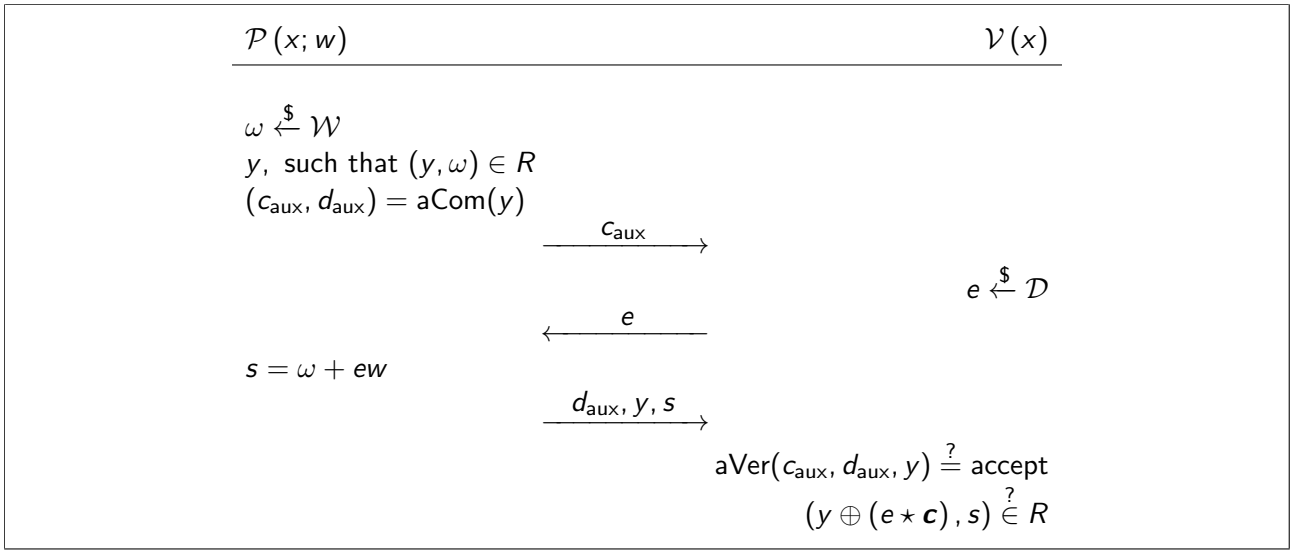
1. $\mathcal{P}$ sends a message a to $\mathcal{V}$
2. $\mathcal{V}$ answers with a random challenge e
3. $\mathcal{P}$ sends an answer z, then $\mathcal{V}$ accepts or rejects the proof checking the conversation (x, a, e, z).

And has the following properties:

- **Completeness**: if an honest prover $\mathcal{P}$ knowns a valid witness w such that $(x, w) \in R$ and follows the protocol, then an honest verifier $\mathcal{V}$ always accepts the conversation.
- **Soundness**: from any pair of accepted conversations (x, a, e, z), (x, a, e', z'), with $e \neq e'$, it is possible to efficiently extract a witness w such that $(x, w) \in R$.
- **Zero-Knowledge**: there exists a polynomial time simulator that takes as input x and a random e and outputs an accepted conversation (x, a, e, z) with the same probability distribution as conversations between honest $\mathcal{P}$ and $\mathcal{V}$.

The usual structure for a Σ-protocol is the following:

Protocol 1: General Σ-proof.



$\oplus$ and $\star$ should be operations compatible with the relation R, and s should be a witness of $y \oplus (e \star \boldsymbol{c})$. This is possible for many relations with small changes in the protocol. If the prover is honest a witness could be extracted from $s = \omega + ew$ and $s' = \omega + e'w$ computing $w = (s - s') / (e - e')$. For each particular protocol it should be proven that this extracted witness is indeed a valid witness, without relying on the honesty of the prover.

The zero-knowledge property can be relaxed to only require that the probability distributions are statistically close, or are computationally indistinguishable, getting statistical zero-knowledge or computational zero-knowledge.

It is clear that the completeness property is required if we want the protocol to be useful. Soundness property may seem more obscure, but it is only bounding the probability of a false positive. If the verifier is honest the probability of a false positive is at most 1 over the number of challenges (which is usually exponentially large). Once the commitment has been sent a dishonest prover may know how to answer to one of the possible challenges, but not two of them, because in that case he would be able to compute a witness, and would not be dishonest. Finally the zero-knowledge property tell us that a dishonest prover can fool a verifier with probability exactly 1 over the number of challenges, by generating a simulated conversation and hoping the verifier sends the same challenge he has chosen. The existence of a simulator implies that a valid conversation reveals no relevant information, as similar conversations can be simulated by anyone.

Observation 2.8. Soundness comes from a probability over the choice of the challenge, once the commitment has been fixed, and for this reason the conversation proves something only if it has been done interactively in the correct order.

However a proof of this kind only convinces the verifier, who is the only person knowing that he has chosen the challenge randomly after the initial commitment was send. To avoid repeating the proof to many verifiers usually this interactive protocol is transformed into a non interactive zero-knowledge proof of knowledge via the Fiat-Shamir transformation [FS86]. In the Fiat-Shamir transformation only the prover plays a role, and there is no specific verifier. The random challenge is then replaced by a hash of the

commitment, trying to emulate the random behavior of the verifier.

Definition 2.9 (Random Oracle)**.** A *random oracle* $\mathcal{O} : A \to B$ takes queries from A and outputs uniformly random elements from B, keeping a table of all the previous queries so that if the query has been already answered it outputs the same element.

The Fiat-Shamir transform is sound in the Random Oracle Model (ROM), where hash functions are assumed to behave as Random Oracles. However it is not clear how this model could be used in a quantum scenario. It has been proved that it is quantumly insecure under assumptions that are sufficient for classical security [ARU14]. The Quantum Random Oracle Model (QROM) is defined as the ROM but allowing queries to be a superposition of elements, answered with a superposition of random elements.

Another problem is the use of rewinding in the Fiat-Shamir transformation. A witness could be extracted if once the computation has ended we rewind the Turing Machine to the point before the oracle has been called, then changing the oracle we could get a different challenge, run it again and extract the witness from the two accepted computations. Quantum computers can not be rewinded as a Turing Machine, all computations are invertible, but we do not know the state of the qubits until we perform a measurement, and this operation is physically non-rewindable, as the measurement disturbs the state. This same problem arises when considering the queries to the quantum oracle, that can not be measured.

Unruh proposed in [Unr15] the first and only transformation from an Interactive zero-knowledge Proof to a Non Interactive Zero-Knowledge Proof secure in a quantum scenario. In his proposal the prover computes several commitments, many challenges for each of them and their corresponding answers. Commitments, challenges and hashes of answers are published. Then a second hash function is used to determine which of the challenges to each commitment should be answered, and its corresponding answer is published. The verifier checks that published conversations are valid and all hashes have been applied correctly (the answers published when hashed correspond to the pretended hashes, and the answers revealed correspond to the ones indicated by the second hash function).

$$\forall i \in I, \quad \forall j \in J$$

$\mathcal{P}$ computes valid conversations:

$$(x, a_i, e_{i,j}, z_{i,j})$$

$\mathcal{P}$ reveals:

$$(x, a_i, e_{i,j}, H_1(z_{i,j}))$$

$\mathcal{P}$ computes a hash of all of this to decide which answers to reveal:

$$\boldsymbol{h} = H_2\left((x, a_i, e_{i,j}, H_1(z_{i,j}))_{i,j}\right) \in J^{|I|}$$

$\mathcal{P}$ finally reveals those answers:

$$\forall i \in I$$

$$(x, a_i, e_{i,h_i}, z_{i,h_i})$$

This more intricate procedure allows to prove soundness and zero-knowledge in a quantum scenario, and can be used to transform all our Σ-protocols into Non Interactive Proofs.

2.4 Lattices

2.4.1 Introduction to lattices

Definition 2.10 (Lattice)**.** A *lattice* $\mathcal{L}$ is a set of points in an n-dimensional space, usually $\mathbb{R}^n$, with a periodic structure. That is, the following two conditions hold:

- It is an additive subgroup: $0 \in \mathcal{L} \quad$ and $\quad \forall x, y \in \mathcal{L} \quad -x, x+y \in \mathcal{L}$
- It is discrete: $\forall x \in \mathcal{L}$ there exists a neighborhood of x in $\mathbb{R}^n$ such that x is the only point of the lattice.

Usually a lattice is defined by a basis of vectors.

Definition 2.11 (Generated lattice)**.** Given k linearly independent vectors $\boldsymbol{b}_1, \dots, \boldsymbol{b}_k \in \mathbb{R}^n$, the *lattice generated* by them is the following set:

$$\mathcal{L}(\boldsymbol{b}_1, \dots, \boldsymbol{b}_k) = \left\{ \sum_{i=1}^{k} z_i \boldsymbol{b}_i \;\middle|\; z_i \in \mathbb{Z} \right\} = \{ \boldsymbol{B}\boldsymbol{z} \mid z \in \mathbb{Z}^n \} = \mathcal{L}(\boldsymbol{B})$$

We have called $\boldsymbol{B}$ to the matrix whose columns are vectors $\boldsymbol{b}_i$. We say $\boldsymbol{b}_1, \dots, \boldsymbol{b}_k$ form a *basis* of the lattice $\mathcal{L}(\boldsymbol{B})$.

A lattice can be defined by different basis. Multiplying $\boldsymbol{B}$ by a unimodular matrix $\boldsymbol{U}$ gives us another basis for the same lattice, in fact all basis for this lattice can be obtained this way.

Theorem 2.12. *$\mathcal{L}(\boldsymbol{B}) = \mathcal{L}(\boldsymbol{B'})$ if and only if there exist a unimodular matrix $\boldsymbol{U}$ such that $\boldsymbol{B'} = \boldsymbol{B}\boldsymbol{U}$.*

Proof. Assume $\boldsymbol{B'} = \boldsymbol{B}\boldsymbol{U}$. As $\boldsymbol{U}$ is unimodular its inverse $\boldsymbol{U}^{-1}$ is unimodular too. In particular they are both integer matrices and we also have $\boldsymbol{B} = \boldsymbol{B'}\boldsymbol{U}^{-1}$. Columns of $\boldsymbol{B'}$ are integer combinations of columns of $\boldsymbol{B}$ ($\mathcal{L}(B') \subset \mathcal{L}(B)$) and columns of $\boldsymbol{B}$ are integer combinations of columns of $\boldsymbol{B'}$ ($\mathcal{L}(B) \subset \mathcal{L}(B')$). Therefore $\mathcal{L}(\boldsymbol{B}) = \mathcal{L}(\boldsymbol{B'})$.

Assume $\mathcal{L}(\boldsymbol{B}) = \mathcal{L}(\boldsymbol{B'})$. Then each colum $\boldsymbol{b'_i}$ of $\boldsymbol{B'}$ is a point of the lattice $\mathcal{L}(\boldsymbol{B})$, generated by the columns of $\boldsymbol{B}$. Then $\boldsymbol{b'_i} = \boldsymbol{B}\boldsymbol{u_i}$, where $\boldsymbol{u_i} \in \mathbb{Z}^k$ are the corresponding coefficients of the columns of $\boldsymbol{B}$. We have $\boldsymbol{B'} = \boldsymbol{B}\boldsymbol{U}$ where $\boldsymbol{U}$ is an integer matrix. And by the same argument $\boldsymbol{B} = \boldsymbol{B'}\boldsymbol{V}$, where $\boldsymbol{V}$ is another integer matrix. Combining the two expressions we get $\boldsymbol{B'} = \boldsymbol{B'}\boldsymbol{V}\boldsymbol{U}$. Then $\boldsymbol{B'}(\boldsymbol{V}\boldsymbol{U} - \boldsymbol{Id}) = \boldsymbol{0}$. Since $\boldsymbol{B'}$ is non singular $\boldsymbol{V}\boldsymbol{U} = \boldsymbol{Id}$ and $\boldsymbol{U}$ is unimodular. □

This is important as some problems over lattices are hard or easy depending on some properties of the basis defining the lattice that is given as input. Informally, a basis is said to be *good* if it has short highly orthogonal vectors, and *bad* if it has low orthogonality. Some algorithms work well with highly orthogonal basis, but have a very low probability of success if the basis has low orthogonality. We also say that a highly orthogonal basis has a low orthogonality defect and vice versa.

Definition 2.13 (Orthogonality defect)**.** The *orthogonality defect* of a lattice basis $\boldsymbol{B}$ is given by:

$$\delta(\boldsymbol{B}) = \frac{\prod_{i=1}^{N} \|\boldsymbol{b}_i\|}{\det(\boldsymbol{B})}$$

It can also be normalized taking the n-root: $\sqrt[n]{\delta(\boldsymbol{B})}$.

Definition 2.14 (minimum λ_i)**.** We define the *minimum* $\lambda_i(\mathcal{L})$ as the radius of the smallest hypersphere centered in the origin that contains at least i linearly independent points of the lattice $\mathcal{L}$.

Nevertheless, transforming a *bad* basis into a *good* basis, with short orthogonal vectors, is a difficult problem. In a two dimensional lattice it can be done performing gaussian elimination. This can be generalized to an n-dimensional lattice as it is done in the Lenstra-Lenstra-Lovasz (LLL) algorithm [LLL82], performing gaussian elimination on the elements of the basis two by two. However the obtained basis has vectors whose length is still exponentially far from optimal.

This can be improved working with blocks of k vectors, instead of pairs of vectors, as it is done in the Blockwise Korkine-Zolotarev (BKZ) reduction [SE94]. This algorithm has to find the shortest vector of a k-dimensional lattice as a subroutine, which is again a hard problem (known as SVP) even approximately, as will be explained latter in this section.

The last improvements on finding algorithms able to obtain better basis focus on this k-dimensional SVP problem, following two different approaches. Enumeration techniques [FP85, Kan83, GNR] try to do an exhaustive search, and intend to achieve better efficiency without decreasing so much its performance cleverly reducing its search space. Sieving techniques [NV08, WLTB11] sample random vectors of the lattice using small integer linear combinations of the vectors of the basis and tries to find pairs of them that are close (if two vectors are close its difference is a short vector of the lattice). In order to improve efficiency not all differences are computed, the space is divided into several regions and only sampled lattice points of the same region are compared. Those regions can be delimited by hyperplanes, cones centered at the origin, hyperspheres centered at randomly chosen lattice points, etc. Those different possibilities yield up to a great variety of algorithms with different performance and computational cost (regarding time and space).

However, despite all those recent techniques and improvements their computational cost is still exponential, and some of the algorithms that have better asymptotic cost due to a smaller constant in the exponent also have worse behavior for smaller dimensions, and the asymptotic improvement only dominates for dimensions far above what it is used for cryptography.

In order to work with lattices in a computer is preferable to work modulo a prime q. A q-ary lattice is an integer lattice where the belonging of a point x to the lattice is determined by $x \mod q$.

Definition 2.15 (q-ary lattices)**.** A lattice $\mathcal{L}$ is said to be *q-ary* if $q\mathbb{Z}^n \subseteq \mathcal{L} \subseteq \mathbb{Z}^n$, for an integer q.

Definition 2.16 (Dual Lattice)**.** The *dual lattice* of $\mathcal{L} \subset \mathbb{R}^n$ is $\mathcal{L}^* := \{\boldsymbol{w} | \langle \boldsymbol{w}, \mathcal{L} \rangle \subseteq \mathbb{Z}\}$.

There are two usual ways of representing a q-ary lattice given a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$. The first one is called the Λ_q form:

$$\Lambda_q(\boldsymbol{A}) = \{\boldsymbol{y} \in \mathbb{Z}^n \quad | \quad \boldsymbol{y} = \boldsymbol{A}\boldsymbol{z} \mod q : \boldsymbol{z} \in \mathbb{Z}^m\} \tag{1}$$

And the other is called the orthogonal Λ_q form:

$$\Lambda_q^{\perp}(\boldsymbol{A}) = \left\{\boldsymbol{y} \in \mathbb{Z}^n \quad \middle| \quad \boldsymbol{A}^T\boldsymbol{y} = 0 \mod q\right\} \tag{2}$$

Proposition 2.17. *Lattices $\Lambda_q(\boldsymbol{A})$ and $\Lambda_q^{\perp}(\boldsymbol{A})$ are dual of each other, up to normalization. $\Lambda_q^{\perp}(\boldsymbol{A}) = q\Lambda_q(\boldsymbol{A})^*$ and $\Lambda_q(\boldsymbol{A}) = q\Lambda_q^{\perp}(\boldsymbol{A})^*$.*

Proof.

$\Lambda_q^\perp(\boldsymbol{A}) \subseteq q\Lambda_q(\boldsymbol{A})^*$

$\boldsymbol{y} \in \Lambda_q^\perp(\boldsymbol{A}) \subseteq \mathbb{Z}^n$

$\boldsymbol{y}^T\boldsymbol{A} = 0 \mod q$

$\boldsymbol{y}^T\boldsymbol{A} = q\boldsymbol{a}^T, \quad \boldsymbol{a} \in \mathbb{Z}^m$

$(q^{-1}\boldsymbol{y})^T\boldsymbol{A} = \boldsymbol{a}^T$

$(q^{-1}\boldsymbol{y})^T(\boldsymbol{A}\boldsymbol{z}) = \boldsymbol{a}^T\boldsymbol{z} \in \mathbb{Z}$

$(q^{-1}\boldsymbol{y}) \in \Lambda_q(\boldsymbol{A})^*$

$\boldsymbol{y} \in q\Lambda_q(\boldsymbol{A})^*$

$\implies \Lambda_q^\perp(\boldsymbol{A}) \subseteq q\Lambda_q(\boldsymbol{A})^*$

$\Lambda_q^\perp(\boldsymbol{A}) \supseteq q\Lambda_q(\boldsymbol{A})^*$

$\boldsymbol{y} \in q\Lambda_q(\boldsymbol{A})^*$

$\boldsymbol{y} = q\boldsymbol{y'}, \quad \boldsymbol{y'} \in \Lambda_q(\boldsymbol{A})^*$

$\boldsymbol{y}^T\boldsymbol{A} = q\boldsymbol{y'}^T\boldsymbol{A}$

$\boldsymbol{y}^T\boldsymbol{A} = q\boldsymbol{a}, \quad \boldsymbol{a} \in \mathbb{Z}^m$

$\boldsymbol{y}^T\boldsymbol{A} = 0 \mod q$

$\boldsymbol{y} \in \Lambda_q^\perp(\boldsymbol{A})$

$\implies q\Lambda_q(\boldsymbol{A})^* \subseteq \Lambda_q^\perp(\boldsymbol{A})$

From the definition of the dual lattice it is easy to see that the dual of a dual lattice is the original lattice itself $(\mathcal{L}^*)^* = \mathcal{L}$ (this can be proved finding a basis of the dual lattice in terms of a basis of the original lattice) and that $(q\mathcal{L})^* = q^{-1}\mathcal{L}^*$ (direct consequence of the definition). Applying those two properties to the previous equation we get:

$$\left(\Lambda_q^\perp(\boldsymbol{A})\right)^* = (q\Lambda_q(\boldsymbol{A})^*)^*$$
$$q\left(\Lambda_q^\perp(\boldsymbol{A})\right)^* = \Lambda_q(\boldsymbol{A})$$

□

Now we can define some of the problems whose hardness will be used as hypothesis for our cryptographic constructions.

Definition 2.18 (Approximate Shortest Vector Problem (γ-SVP))**.** Given a base $\boldsymbol{B}$ of a lattice $\mathcal{L}(\boldsymbol{B})$ the *Approximate Shortest Vector Problem*, or *γ-SVP*, is to find a non-zero vector $\boldsymbol{v} \in \mathcal{L}(\boldsymbol{B})$ such that $\|\boldsymbol{v}\| \leq \gamma \cdot \lambda_1(\mathcal{L}(\boldsymbol{B}))$.

The difficulty of this problem depends on the approximating factor γ. If $\gamma = 1$ it is called the Shortest Vector Problem (SVP). Our problems involve a $\gamma(n)$ being a polynomial of the dimension of the lattice. It has been proven to be an NP-hard problem in its exact version and also for some subpolynomial approximations [Ajt98]. The best algorithms for the approximated versions have exponential cost [GN08] and it is believed that no PPT algorithm exists.

This same problem is called Short Integer Solution (SIS) when described in terms of the dual lattice. Finding short vectors in a dual lattice $\Lambda_q^\perp(\boldsymbol{A})$ is finding short integer solutions to the linear system $\boldsymbol{A}\boldsymbol{x} = 0$

Definition 2.19 (Approximate Closest Vector Problem (γ-CVP))**.** Given a base $\boldsymbol{B}$ of a lattice $\mathcal{L}(\boldsymbol{B})$ and a target vector $\boldsymbol{t}$, the *Approximate Closest Vector Problem*, or *γ-CVP*, is to find a vector $\boldsymbol{u} \in \mathcal{L}(\boldsymbol{B})$ such that if $\boldsymbol{v} = \underset{\boldsymbol{w} \in \mathcal{L}(\boldsymbol{B})}{\arg\min} \|\boldsymbol{t} - \boldsymbol{w}\|$, then $\|\boldsymbol{u} - \boldsymbol{v}\| \leq \gamma \|\boldsymbol{t} - \boldsymbol{v}\|$.

Finally the main problem we work with is the following:

Definition 2.20 (Learning With Errors (LWE))**.** Let n, q be integers (q usually prime), χ a discrete probability distribution in $\mathbb{Z}$ (usually a discrete Gaussian distribution) and $\boldsymbol{s}$ a secret vector from $\mathbb{Z}_q^n$.

We denote $\mathcal{L}_{\boldsymbol{s},\chi}$ the probability distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing $\boldsymbol{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing $e \xleftarrow{\$} \chi$ and considering it in $\mathbb{Z}_q$ and finally calculating $(\boldsymbol{a}, c = \langle \boldsymbol{a}, \boldsymbol{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.

The *Decisional Learning With Errors* problem, or Decisional-LWE, is to decide if pairs $(\boldsymbol{a}, c)$ are samples of $\mathcal{L}_{\boldsymbol{s},\chi}$ or come from the uniform distribution of $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

The *Search Learning With Errors* problem, or Search-LWE, is to recover $\boldsymbol{s}$ from samples $(\boldsymbol{a}, c)$ obtained from $\mathcal{L}_{\boldsymbol{s},\chi}$.

If the number of samples provided is polynomial those problems are believed to be hard (it is as hard as the approximated *SVP* [LPR13]), even if the secret $\boldsymbol{s}$ is chosen component by component from the error distribution χ [SG16].

Considering all polynomially many samples as rows in a matrix we can see the problem as $(\boldsymbol{A}, \boldsymbol{As} + \boldsymbol{e})$. If we see $\boldsymbol{A}$ as a basis of a lattice then the search problem becomes recovering the coordinates of a lattice point after adding some error $\boldsymbol{e}$, while the decision problem is to distinguish uniformly random points in $\mathbb{Z}^n$ from perturbed lattice points.

If $\boldsymbol{A}$ is a basis of a full rank lattice, i.e., it is a non singular square matrix, then its size is quadratic in the dimension n, and this yields to a high communication cost. It can be reduced to linear cost if we restrict ourselves to a particular class of lattices.

2.4.2 Ideal Lattices

Fixed a vector $\boldsymbol{f}$ we define the transformation matrix $\boldsymbol{F}$ as:

$$\left[\begin{array}{ccc|c} 0 & \dots & 0 & -f_0 \\ \ddots & & & -f_1 \\ & Id_{n-1} & & \vdots \\ & & \ddots & -f_{n-1} \end{array}\right]$$

Definition 2.21 (Ideal Lattice)**.** An *ideal lattice* is a lattice $\mathcal{L}$ that has as a basis a matrix $\boldsymbol{A}$ constructed from a vector $\boldsymbol{a}$ and a transformation matrix $\boldsymbol{F}$ in the following way:

$$\boldsymbol{A} = \left[\boldsymbol{a}, \boldsymbol{Fa}, \dots, \boldsymbol{F}^{n-1}\boldsymbol{a}\right]$$

Those lattices are called ideal lattices as they can be seen as ideals in the polynomial ring $R_q = \mathbb{Z}_q[x] / \langle f(x) \rangle$, where $f(x) = x^n + f_{n-1}x^{n-1} + \cdots + f_0 \in \mathbb{Z}_q[x]$ is a polynomial given by the vector of the transformation matrix. From this point we will identify any other vector $\boldsymbol{v} = (v_1, v_2, \dots, v_n) \in \mathbb{Z}_q^n$ with a polynomial $v = v_1 + v_2 x + \cdots + v_n x^{n-1} \in R_q$. It can be easily checked that, by construction, multiplying two polynomials a and b in the ring R_q is equivalent to multiply the matrix $\boldsymbol{A}$ constructed from the vector $\boldsymbol{a}$ with the vector $\boldsymbol{b}$. Then, lattice points of $\mathcal{L}(\boldsymbol{A})$ are, as polynomials, elements of the principal ideal $\langle a \rangle \subset R_q$.

We choose f to be $(1, 0, \dots, 0)$, so that $f(x) = x^n + 1$, with n a power of 2, and $R_q = \mathbb{Z}_q[x] / \langle x^n + 1 \rangle$, as it gives us good security reductions and allows us to speed up computations using the Fast Fourier Transform.

This way the matrix $\boldsymbol{A}$ is an anti-cyclic integer matrix:

$$\boldsymbol{A} = \begin{pmatrix} a_1 & -a_n & -a_{n-1} & \dots & -a_2 \\ a_2 & a_1 & -a_n & \dots & -a_3 \\ a_3 & a_2 & a_1 & \dots & -a_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_n & a_{n-1} & a_{n-2} & \dots & a_1 \end{pmatrix}$$

No algorithm is known to be able to take advantage of the structure in ideal lattices to efficiently solve the problems presented before when restricted to their ideal cases. In particular we make use of cryptographic primitives based on the ideal version of the learning with errors algorithm, called ring learning with errors, or RLWE.

2.4.3 Encryption with RLWE

Lyubashevsky, Peikert and Regev proposed in [LPR13] the following encryption scheme based on the hardness of RLWE:

- **Gen**: the key generator algorithm outputs $(R_q, \chi_\sigma, sk = s, pk = (a_\mathsf{E}, b_\mathsf{E})) \xleftarrow{\$} \mathtt{Gen}\left(1^\lambda\right)$. $R_q = \mathbb{Z}_q[x] / \langle x^n + 1 \rangle$ is a polynomial ring where n and q have been chosen according to λ. χ_σ is a discretized Gaussian distribution over R_q with standard deviation $\sigma = \alpha q / \sqrt{2\pi}$. $a_\mathsf{E} \xleftarrow{\$} R_q$ is obtained uniformly at random, $s, e \xleftarrow{\$} \chi_\sigma$ are small elements obtained from the error distribution and b_E is computed $b_\mathsf{E} = a_\mathsf{E} \cdot s + e$. That is, the public key is a RLWE sample $(a_\mathsf{E}, b_\mathsf{E})$ and the secret key is the corresponding secret s.
- **Enc**: the encryption algorithm takes as input a message $m \in \{0,1\}^n$ encoded as a polynomial in R_q with 0 or 1 coefficients and a public key $(a_\mathsf{E}, b_\mathsf{E})$. It chooses small random elements $r_\mathsf{E}, e_u, e_v \xleftarrow{\$} \chi_\sigma$ and computes the ciphertext $(u, v) = \left(a_\mathsf{E} \cdot r_\mathsf{E} + e_u, b_\mathsf{E} \cdot r_\mathsf{E} + e_v + \left\lfloor \frac{q}{2} \right\rceil m\right) \in R_q \times R_q$.
- **Dec**: the decryption algorithm takes as input a ciphertext (u, v) and a secret key s and computes:

$$\begin{aligned} v - s \cdot u &= b_\mathsf{E} \cdot r_\mathsf{E} + e_v + \left\lfloor \frac{q}{2} \right\rceil m - s(a_\mathsf{E} \cdot r_\mathsf{E} + e_u) \\ &= s \cdot a_\mathsf{E} \cdot r_\mathsf{E} + e \cdot r_\mathsf{E} + e_v + \left\lfloor \frac{q}{2} \right\rceil m - s \cdot a_\mathsf{E} \cdot r_\mathsf{E} - s \cdot e_u \\ &= e \cdot r_\mathsf{E} + e_v - s \cdot e_u + \left\lfloor \frac{q}{2} \right\rceil m \\ &\approx \left\lfloor \frac{q}{2} \right\rceil m \end{aligned}$$

If the standard deviation σ of the distribution error is small enough compared with $\left\lfloor \frac{q}{2} \right\rceil$ we can recover the message m with overwhelming probability rounding the coefficients of the polynomial to either 0 or $\left\lfloor \frac{q}{2} \right\rceil$, whichever is closer $\mod q$. We also introduce the possibility of a re-encryption:

- **Re-Enc**: the re-encryption algorithm takes as input a ciphertext (u, v) and a public key $(a_\mathsf{E}, b_\mathsf{E})$. It choses small random elements $r'_\mathsf{E}, e'_u, e'_v \xleftarrow{\$} \chi_\sigma$ from the error distribution and computes the re-encrypted ciphertext $\mathtt{Re-Enc}(u, v) = (u + a_\mathsf{E} \cdot r'_\mathsf{E} + e'_u, v + b_\mathsf{E} \cdot r'_\mathsf{E} + e'_v) \in R_q \times R_q$.

When decrypting a re-encrypted ciphertext we obtain: $e \cdot (r_E + r'_E) + (e_v + e'_v) - s \cdot (e_u + e'_u) + \lfloor \frac{q}{2} \rceil m$. The errors may grow linearly, so only a limited number of re-encryptions can be performed if we want to keep a high probability of decrypting without errors. This is not a problem, as our protocols perform a fixed small number of re-encryptions, so that the parameters σ and q can be selected taking it into account.

Theorem 2.22. *The above cryptosystem is IND-CPA secure assuming the hardness of decision-RLWE.*

Proof. Assume $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is a PPT adversary such that the following quantity is non negligible in λ:

$$\left| \Pr \left[b' = b \;\middle|\; \begin{array}{c} (s, (a_E, b_E)) \xleftarrow{\$} \mathtt{Gen}\left(1^\lambda\right), (m_0, m_1, aux) \xleftarrow{\$} \mathcal{A}_1\left((a_E, b_E), 1^\lambda\right), b \xleftarrow{\$} \{0,1\}, \\ (u, v) \xleftarrow{\$} \mathtt{Enc}\left(m_b; pk\right), b' \xleftarrow{\$} \mathcal{A}_2\left((u, v), (a_E, b_E), aux, 1^\lambda\right) \end{array} \right] - \frac{1}{2} \right|$$

Let $(a_1, c_1), (a_2, c_2) \in R_q \times R_q$ be two pairs that can be uniformly random pairs (probability $1/2$) or RLWE samples (probability $1/2$). If they are random polynomials then:

$$\left| \Pr \left[b' = b \;\middle|\; \begin{array}{c} (m_0, m_1, aux) \xleftarrow{\$} \mathcal{A}_1\left((a_1, a_2), 1^\lambda\right), b \xleftarrow{\$} \{0,1\}, \\ (u = c_1, v = c_2 + \lfloor \frac{q}{2} \rceil m_b), b' \xleftarrow{\$} \mathcal{A}_2\left((u, v), (a_1, a_2), aux, 1^\lambda\right) \end{array} \right] - \frac{1}{2} \right| = 0$$

As the uniformly random element c_2 completely masks the chosen message the output b' is independent of b, and the probability of correctly guessing the bit is exactly $1/2$.

If $(a_1, c_1), (a_2, c_2)$ are real RLWE samples there are two possibilities, either the difference with $1/2$ is non negligible (allowing us to solve the decision-RLWE with non negligible probability) or it is negligible. In this second case it means that the adversary $\mathcal{A}$ works well when the elements (a_1, a_2) playing the role of the public key (a_E, b_E) are a real RLWE sample, but not when they are two random polynomials, which also gives us a way to distinguish random pairs from RLWE samples, playing the same game this time choosing a_2 to be the element that can be a uniformly random polynomial or a RLWE sample of the lattice generated by a_1. □

Observation 2.23. Notice that we do not know a priori which of those possibilities would be true, but all of them allow us to distinguish RLWE samples from random elements, using different strategies.

2.4.4 Commitments with RLWE

We use for our constructions the commitment proposed by Benhamouda *et al.* in [BKLP15], as it allows us to commit to a message and furthermore it allows us to prove in zero-knowledge polynomial relations between the messages committed.

It is a commitment scheme perfectly binding with overwhelming probability under the choice of the public key and computationally hiding under the RLWE assumption. This long term privacy is very important for us, as we do not plan to open most of the commitments we construct during our protocols.

In order to achieve the binding property this scheme makes use of vectors of polynomials, that we denote by lowercase boldface roman letters as integer vectors like $\boldsymbol{a} \in (R_q)^k$. It also needs that the prime q is $q \equiv 3 \mod 8$. This implies $x^n + 1$ splits into two irreducible polynomials of degree $n/2$ [BGM93], and every polynomial of degree smaller than $n/2$ is invertible.

The commitment scheme is defined by the following three algorithms:

- **Gen**: the generator algorithm takes a security parameter 1^λ and outputs a public key $pk = (\boldsymbol{a}_C, \boldsymbol{b}_C) \xleftarrow{\$} \texttt{Gen}\left(1^\lambda\right)$, where $\boldsymbol{a}_C, \boldsymbol{b}_C$ are uniformly random vectors of polynomials $\boldsymbol{a}_C, \boldsymbol{b}_C \xleftarrow{\$} (R_q)^k$ and parameters as n or k are defined by $\texttt{Gen}$ acording to λ.

- **Com**: the commitment algorithm takes as input a message m and a public key $(\boldsymbol{a}_C, \boldsymbol{b}_C)$ and produces a commitment $\boldsymbol{c}$ and an opening d. $(\boldsymbol{c} = \boldsymbol{a}_C m + \boldsymbol{b}_C r_C + \boldsymbol{e}_C, d = (m, r_C, \boldsymbol{e}_C, 1)) \xleftarrow{\$} \texttt{Com}(m; pk)$. Polynomial r_C is chosen uniformly at random from R_q and vector of polynomials $\boldsymbol{e}_C$ is obtained from an error distribution χ_{σ_e}.

- **Ver**: the verification algorithm takes as input a commitment $\boldsymbol{c}$, a message with its opening $(m', r'_C, \boldsymbol{e}'_C, f')$, and a public key $(\boldsymbol{a}_C, \boldsymbol{b}_C)$ and accepts if the following conditions hold:

$$\boldsymbol{a}_C m' + \boldsymbol{b}_C r'_C + f'^{-1}\boldsymbol{e}'_C = \boldsymbol{c} \wedge \left\|\boldsymbol{e}'_C\right\|_\infty \leq \left\lfloor \frac{n^{4/3}}{2} \right\rceil \wedge \left\|f'\right\|_\infty \leq 1 \wedge \deg f' \leq \frac{n}{2}$$

This is not a standard commitment scheme as the accepted openings are relaxed from the produced openings (the commitment algorithm always generates $f = 1$ while in the opening it is accepted to have a low degree polynomial with infinity norm 1). That is, we use knowledge of $(x, w) \in R$ to prove knowledge of $(x, w') \in R'$, with $R \subsetneq R'$. This relaxation would be necessary for the following zero-knowledge proofs, but does not compromise the binding property. Benhamouda *et al.* give in [BKLP15] a combinatorial argument showing that even allowing openings with $f \neq 1$ the probability over the choice of the public key of the existence of a commitment with valid openings to two different messages is negligible in the security parameter if all the parameters q, n, k are chosen correctly. The hiding property is clearly based on the hardness of RLWE.

Some commitments schemes, as Pedersen Commitment [Ped91], are homomorphic as a commitment of the sum of two messages can be obtained from the product of the commitments to each message. With Pedersen commitments also the commitment of a scalar multiplied by a message can be computed exponentiating a commitment to the message to that scalar. This commitment scheme is specially interesting as, even if it is not directly homomorphic, allows someone knowing the openings to prove knowledge of a valid opening and polynomial relations between the messages of different commitments. We expose here the protocol presented by Benhamouda *et al.* for proving knowledge of a valid opening, as it uses some ideas that are worth to discuss and also gives us a flavor of the complexity of this kind of proofs.

$$\text{ZK-proof}\left[\, m, r_C, \boldsymbol{e}_C, f \;\middle|\; \text{Ver}(\boldsymbol{c}; m, r, \boldsymbol{e}, f) = \text{accept} \,\right] \tag{3}$$

Protocol 2: Simple preimage proof.

$\mathcal{P}(\boldsymbol{c}; m, r_C, \boldsymbol{e}_C)$		$\mathcal{V}(\boldsymbol{c})$
$\mu, \rho \xleftarrow{\$} \mathbb{Z}_q[x] / \langle x^n + 1 \rangle$		
$\boldsymbol{\eta} \xleftarrow{\$} \chi^k_{\sigma_\eta}$		
$\boldsymbol{t} = \boldsymbol{a}_C \mu + \boldsymbol{b}_C \rho + \boldsymbol{\eta}$		
$(c_{\text{aux}}, d_{\text{aux}}) = \text{aCom}(\boldsymbol{t})$		
	$\xrightarrow{c_{\text{aux}}}$	
		$d \xleftarrow{\$} \mathcal{C} \subseteq \mathbb{Z}_q[x] / \langle x^n + 1 \rangle$
	$\xleftarrow{d}$	
$s_m = \mu + dm$		
$s_r = \rho + d r_C$		
$s_e = \boldsymbol{\eta} + d\boldsymbol{e}_C$		
abort with probability:		
$\exp\left(\frac{-2\langle s_e, d\boldsymbol{e}_C\rangle + \|d\boldsymbol{e}_C\|^2}{2\sigma_\eta^2}\right)$		
	$\xrightarrow[s_m, s_r, s_e]{d_{\text{aux}}, \boldsymbol{t}}$	
		$\text{aVer}(c_{\text{aux}}, d_{\text{aux}}, \boldsymbol{t}) \stackrel{?}{=} \text{accept}$
		$\boldsymbol{t} + d\boldsymbol{c} \stackrel{?}{=} \boldsymbol{a}_C s_m + \boldsymbol{b}_C s_r + s_e$
		$\|s_e\|_\infty \stackrel{?}{\leq} \lfloor n^{4/3}/4 \rfloor$

It follows the usual scheme of a zero-knowledge proof described in protocol 1. Given $\boldsymbol{c}$, to prove knowledge of $m, r_C, \boldsymbol{e}_C$ the prover $\mathcal{P}$ builds $\boldsymbol{t}$ following the same structure with random masking parameters $\mu, \rho, \boldsymbol{\eta}$. It commits to $\boldsymbol{t}$ using an auxiliary commitment scheme, receives a challenge d and then builds $\boldsymbol{t} + d\boldsymbol{c}$, reveals its opening and the verifier $\mathcal{V}$ checks that everything follows the expected restrictions. However the possible extracted witness has a factor $(d - d')^{-1}$, this is generally not important, but our commitment scheme requires the $\boldsymbol{e}$ parameter to be small, while we get a small vector multiplied by the inverse of a difference of polynomials. That is why Benhamouda *et al.* designed in the first place its commitment allowing this f^{-1}, and this protocol chooses challenges d from the subset $\mathcal{C} = \{d \in \{0,1\}^n : \|d\|_1 \leq \kappa \wedge \deg d \leq n/2\}$, so that the differences $(d - d')$ may be a valid f. The property of $q \equiv 3 \mod 8$ also ensures that this f is invertible.

The other main difference is that, as ρ is uniformly random then $s_r = \rho + d r_C$ is also uniformly random and reveals no information about r_C. However $\boldsymbol{\eta}$ is not uniformly random, as it is required to follow a distribution $\chi^k_{\sigma_\eta}$. Then $s_e = \boldsymbol{\eta} + d\boldsymbol{e}_C$ reveals some information about $\boldsymbol{e}_C$, as it is a discrete Gaussian distribution centered at $d\boldsymbol{e}_C$. To correct this the protocol aborts with a probability of $\exp\left(\frac{-2\langle s_e, d\boldsymbol{e}_C\rangle + \|d\boldsymbol{e}_C\|^2}{2\sigma_\eta^2}\right)$, implying that the distribution of revealed s_e is statistically indistinguishable from a discrete Gaussian distribution centered at the origin.

This technique, called Fiat-Shamir with aborts, provides statistically zero-knowledge property because of the following theorem:

Theorem 2.24 (Vadim Lyubashevsky [Lyu12]). *Let V be a subset of $\mathbb{Z}^l$ in which all elements have norms less than T, and let h be a probability distribution over V. Then, for any constant M, there exists a $\sigma = \widetilde{\Theta}(T)$ such that the output distribution of the following algorithms A, F are statistically close.*

A :	F :
$\boldsymbol{v} \xleftarrow{\$} h; \quad \boldsymbol{z} \xleftarrow{\$} \chi^l_{\boldsymbol{v},\sigma};$	$\boldsymbol{v} \xleftarrow{\$} h; \quad \boldsymbol{z} \xleftarrow{\$} \chi^l_{\sigma};$
output $(\boldsymbol{z}, \boldsymbol{v})$ *with probability:*	*output* $(\boldsymbol{z}, \boldsymbol{v})$ *with probability:*
$\min\left(\exp\left(\frac{-2\langle \boldsymbol{z}, \boldsymbol{v}\rangle + \|\boldsymbol{v}\|^2}{2\sigma^2}\right), 1\right)$	$\frac{1}{M}$

Moreover the probability that A outputs something is exponentially close to $1/M$.

With those aborts the probability distribution of revealed $s_{\boldsymbol{e}}$ is exponentially close to a probability distribution centered in the origin and non dependent on $d\boldsymbol{e}$. The expected number of iterations until there is no abort is a constant M, that depends on σ.

This protocol can be adapted to prove that a commitment contains a specific linear combination of the messages committed in other commitments (protocol 3).

It follows the same strategy, to prove that commitments $\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3$ contains messages such that $m_3 = \lambda_1 m_1 + \lambda_2 m_2$ it creates masking parameters for $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ and uses a linear combination of them with coefficients λ_1 and λ_2 to build the masking parameters of $\boldsymbol{c}_3$. We can prove linear relations as large as we want, but the probability of abortion increases with each term. For practical purposes it will be implemented adding all the terms consecutively one by one, and providing a proof for each of these partial sums.

It is also possible to prove knowledge of multiplicative relations, with protocol 4. This time we have to take into account that multiplying the commitments we get cross terms, and we have to create masking parameters for them.

Those protocols are the ones proposed by Benhamouda *et al.* (or direct generalizations) in [BKLP15] and the details of the proofs can be found there. In order to prove arbitrary polynomial relations we will need to run several instances of protocols 3 and 4.

$$\text{ZK-proof}\left[m_i, r_i, \boldsymbol{e}_i, f_i \;\middle|\; \begin{array}{c} (m_l = \lambda_1 m_1 + \cdots + \lambda_{l-1} m_{l-1}) \\ \bigwedge_{i=1}^{l} (\text{Ver}(c_i; m_i, r_i, \boldsymbol{e}_i, f_i) = \text{accept}) \end{array} \right] \tag{4}$$

Protocol 3: Linear relation proof.

$\mathcal{P}(\boldsymbol{c}_i; m_i, r_i, e_i)$	$\mathcal{V}(\boldsymbol{c}_i)$
$\mu_1, \ldots, \mu_{l-1} \xleftarrow{\$} R_q$	
$\rho_1, \ldots, \rho_l \xleftarrow{\$} R_q$	
$\mu_l = \lambda_1 \mu_1 + \cdots + \lambda_{l-1} \mu_{l-1}$	
$\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_l \xleftarrow{\$} \chi^k_{\sigma_\eta}$	
for $i = 1, \ldots, l$	
$\boldsymbol{t}_i = \boldsymbol{a}\mu_i + \boldsymbol{b}\rho_i + \boldsymbol{\eta}_i$	
$(c_{\text{aux}}, d_{\text{aux}}) = \text{aCom}(\boldsymbol{t}_1, \ldots, \boldsymbol{t}_l)$	
$\xrightarrow{c_{\text{aux}}}$	
	$d \xleftarrow{\$} \mathcal{C}$
$\xleftarrow{d}$	
for $i = 1, \ldots, l$	
$s_{m_i} = \mu_i + d m_i$	
$s_{r_i} = \rho_i + d r_i$	
$s_{e_i} = \boldsymbol{\eta}_i + d \boldsymbol{e}_i$	
abort with probability:	
$\prod_{i=1}^{l} e^{\left(\frac{-2\langle s_{e_i}, d e_i \rangle + \lVert d e_i \rVert^2}{2\sigma_\eta^2} \right)}$	
$\xrightarrow[s_{m_i}, s_{r_i}, s_{e_i}]{d_{\text{aux}}, \boldsymbol{t}_i}$	
	$\text{aVer}(c_{\text{aux}}, d_{\text{aux}}, (\boldsymbol{t}_1, \ldots, \boldsymbol{t}_l)) \stackrel{?}{=} \text{accept}$
	$s_{m_l} = \lambda_1 s_{m_1} + \cdots + \lambda_{l-1} s_{m_{l-1}}$
	for $i = 1, \ldots, l$
	$\boldsymbol{t}_i + d\boldsymbol{c}_i \stackrel{?}{=} \boldsymbol{a} s_{m_i} + \boldsymbol{b} s_{r_i} + s_{\boldsymbol{e}_i}$
	$\lVert s_{\boldsymbol{e}_i} \rVert \stackrel{?}{\leq} \left\lfloor n^{4/3}/4 \right\rfloor$

$$\text{ZK-proof}\left[m_i, r_i, \boldsymbol{e}_i, f_i \;\middle|\; (m_3 = m_1 m_2) \wedge \bigwedge_{i=1}^{3} (\text{Ver}(c_i; m_i, r_i, \boldsymbol{e}_i, f_i) = \text{accept}) \right] \tag{5}$$

Protocol 4: Multiplicative relation proof.

$\mathcal{P}(\boldsymbol{c}_i; m_i, r_i, e_i)$		$\mathcal{V}(\boldsymbol{c}_i)$
$\mu_1, \mu_2, \mu_3 \xleftarrow{\$} R_q$		
$\rho_1, \rho_2, \rho_3 \xleftarrow{\$} R_q$		
$\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, \boldsymbol{\eta}_3 \xleftarrow{\$} \chi^k_{\sigma_\eta}$		
for $i = 1, 2, 3$		
$\boldsymbol{t}_i = \boldsymbol{a}\mu_i + \boldsymbol{b}\rho_i + \boldsymbol{\eta}_i$		
$m_+ = \mu_1 m_2 + \mu_2 m_1$		
$m_\times = \mu_1 \mu_2$		
$r_+, r_\times \xleftarrow{\$} R_q$		
$\boldsymbol{e}_+, \boldsymbol{e}_\times \xleftarrow{\$} \chi^k_{\sigma_{\boldsymbol{e}}}$		
$\boldsymbol{c}_+ = \boldsymbol{a}m_+ + \boldsymbol{b}r_+ + \boldsymbol{e}_+$		
$\boldsymbol{c}_\times = \boldsymbol{a}m_\times + \boldsymbol{b}r_\times + \boldsymbol{e}_\times$		
$\mu_+, \mu_\times, \rho_+, \rho_\times \xleftarrow{\$} R_q$		
$\boldsymbol{\eta}_+, \boldsymbol{\eta}_\times \xleftarrow{\$} \chi^k_{\sigma_\eta}$		
$\boldsymbol{t}_+ = \boldsymbol{a}\mu_+ + \boldsymbol{b}\rho_+ + \boldsymbol{\eta}_+$		
$\boldsymbol{t}_\times = \boldsymbol{a}\mu_\times + \boldsymbol{b}\rho_\times + \boldsymbol{\eta}_\times$		
$\widetilde{\rho} \xleftarrow{\$} R_q$		
$\widetilde{\boldsymbol{\eta}} \xleftarrow{\$} \chi^k_{\sigma_\eta}$		
$\widetilde{\boldsymbol{t}} = \boldsymbol{b}\widetilde{\rho} + \widetilde{\boldsymbol{\eta}}$		
$(c_{\text{aux}}, d_{\text{aux}}) = \text{aCom}\left(\begin{smallmatrix}\boldsymbol{t}_+, \boldsymbol{t}_\times, \boldsymbol{t}_i, \widetilde{\boldsymbol{t}}, \\ \boldsymbol{c}_+, \boldsymbol{c}_\times\end{smallmatrix}\right)$		
	$\xrightarrow{c_{\text{aux}}}$	
	$\xleftarrow{d}$	$d \xleftarrow{\$} \mathcal{C}$
for $i = 1, 2, 3, +, \times$		
$s_{m_i} = \mu_i + d m_i$		
$s_{r_i} = \rho_i + d r_i$		
$s_{e_i} = \boldsymbol{\eta}_i + d\boldsymbol{e}_i$		
$\widetilde{\boldsymbol{e}} = -d^2\boldsymbol{e}_3 - \boldsymbol{e}_\times - d\boldsymbol{e}_+$		
$\widetilde{r} = -d^2 r_3 - r_\times - d r_+$		
$s_{\widetilde{\boldsymbol{e}}} = \widetilde{\boldsymbol{\eta}} + d\widetilde{\boldsymbol{e}}$		
$s_{\widetilde{r}} = \widetilde{\rho} + d\widetilde{r}$		
abort-cheks for $s_{\widetilde{\boldsymbol{e}}}, s_{\boldsymbol{e}_j}$	$d_{\text{aux}}, \underset{s_{m_i}, s_{r_i}, s_{\boldsymbol{e}_i}, s_{\widetilde{r}}, s_{\widetilde{\boldsymbol{e}}}}{\xrightarrow{\boldsymbol{t}_+, \boldsymbol{t}_\times, \boldsymbol{t}_i, \widetilde{\boldsymbol{t}}, \boldsymbol{c}_+, \boldsymbol{c}_\times}}$	
		$\text{aVer}(c_{\text{aux}}, d_{\text{aux}}, (\boldsymbol{t}_+, \boldsymbol{t}_\times, \boldsymbol{t}_i, \widetilde{\boldsymbol{t}}, \boldsymbol{c}_+, \boldsymbol{c}_\times)) \stackrel{?}{=} \text{accept}$
		for $i = 1, 2, 3, +, \times$
		$\boldsymbol{t}_i + d\boldsymbol{c}_i \stackrel{?}{=} \boldsymbol{a}s_{m_i} + \boldsymbol{b}s_{r_i} + s_{\boldsymbol{e}_i}$
		$\|s_{\boldsymbol{e}_i}\| \stackrel{?}{\leq} \left\lfloor n^{4/3}/4 \right\rfloor$
		$\widetilde{\boldsymbol{c}} = \boldsymbol{a}s_{m_1}s_{m_2} - d^2\boldsymbol{c}_3 - \boldsymbol{c}_\times - d\boldsymbol{c}_+$
		$\widetilde{\boldsymbol{t}} + d\widetilde{\boldsymbol{c}} \stackrel{?}{=} \boldsymbol{b}s_{\widetilde{r}} + s_{\widetilde{\boldsymbol{e}}}$
		$\|s_{\widetilde{\boldsymbol{e}}}\| \stackrel{?}{\leq} \left\lfloor n^{4/3}/4 \right\rfloor$

2.4.5 Trapdoors with lattices

Generally backdoors are something to avoid. For example in case of a commitment scheme the security proof for the hiding property states that it is hard to recover the message with the assumption that the public key has been chosen randomly. However, for some particular specially generated public keys there may be some information that allows a malicious entity to recover the message looking at the commitment.

Because of this, when a public key has to be generated at random, in order to prove that it is really random and not fixed by the generator, it is obtained from a seed with an extendable output function, that guarantees that there is no special property that might be used as a backdoor.

In our case we propose to use SHA-3 standard extendable output functions (XOF) defined in [Dwo15] to avoid backdoors, but we also require the use of backdoors to construct more complex protocols with the required properties. In order to build a backdoor hash we will need a lattice based trapdoor function.

Peikert and Micciancio explain in [MP12] how to build lattice based trapdoor functions for the following families of functions:

$$\begin{aligned} f_A \colon D &\to \mathbb{Z}_q^n \\ \boldsymbol{x} &\mapsto A\boldsymbol{x} \end{aligned}$$

$$\begin{aligned} g_A \colon \mathbb{Z}_q^n &\to \mathbb{Z}_q^m \\ \boldsymbol{x} &\mapsto A^\intercal \boldsymbol{x} + \boldsymbol{e} \end{aligned}$$

D is a subset of *small vectors*, and $\boldsymbol{e}$ is obtained from a discrete gaussian distribution and also has *small* norm. For someone without the trapdoor it is hard to find collisions. An adversary $\mathcal{A}$ able to find a collision could solve some hard problems.

$$\begin{aligned} &\boldsymbol{x}, \boldsymbol{y} \leftarrow \mathcal{A} \text{ s.t. } f_A(\boldsymbol{x}) = f_A(\boldsymbol{y}), \boldsymbol{x}, \boldsymbol{y} \in D \\ &\quad \implies A\boldsymbol{x} = A\boldsymbol{y} \implies A(\boldsymbol{x} - \boldsymbol{y}) = 0 \qquad \text{(solve SIS)} \end{aligned}$$

$$\begin{aligned} &\boldsymbol{x}, \boldsymbol{y} \leftarrow \mathcal{A} \text{ s.t. } g_A(\boldsymbol{x}) = g_A(\boldsymbol{y}), \boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_q^n \\ &\quad \implies A^\intercal \boldsymbol{x} + \boldsymbol{e}_{\boldsymbol{x}} = A^\intercal \boldsymbol{y} + \boldsymbol{e}_{\boldsymbol{y}} \\ &\quad \implies A^\intercal (\boldsymbol{x} - \boldsymbol{y}) = (\boldsymbol{e}_{\boldsymbol{y}} - \boldsymbol{e}_{\boldsymbol{x}}) \qquad \text{(solve SVP)} \end{aligned}$$

We focus on the first one. We start with $\boldsymbol{G}$ (built as a tensor product of an identity matrix and a primitive vector, the details of this construction are explained in [MP12]) a primitive public matrix for which we know how to calculate preimages of $f_{\boldsymbol{G}}$. We extend it to a semirandom matrix $\boldsymbol{A}' = \left[\overline{\boldsymbol{A}} \middle| \boldsymbol{G}\right]$, concatenating it with a uniformly random matrix $\overline{\boldsymbol{A}}$.

$$f_{A'}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \left[\overline{A}\middle|G\right]\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \overline{A}x_1 + Gx_2 = y$$
$$f_{A'}^{-1}(y) = \left({x_1}^\intercal \middle| f_G^{-1}\left(y - \overline{A}x_1\right)^\intercal\right)^\intercal \qquad (x_1 \text{ randomly chosen from a suitable distribution})$$

Finally we multiply it by a transformation matrix $T = \begin{pmatrix} I & -R \\ 0 & I \end{pmatrix}$. We observe that its inverse is $T^{-1} = \begin{pmatrix} I & R \\ 0 & I \end{pmatrix}$.

$$A = A' \cdot T = \left[\overline{A}\middle|G\right]\begin{pmatrix} I & -R \\ 0 & I \end{pmatrix} = \left[\overline{A}\middle|G - \overline{A}R\right]$$

We can invert it into two steps, first inverting A' and then inverting T. Here we take advantage of the fact that, if coefficients of R are small then the coefficients of the inverse of T, are small too, and we still get small solutions:

$$f_A(x) = A'Tx$$
$$f_A^{-1}(y) = T^{-1}f_{A'}^{-1}(y)$$

The final matrix A is pseudorandom and it is difficult to invert without knowing the trapdoor R, while knowledge of R allows to easily obtain preimages of f_A. Peikert and Micciancio give more details of the security of this construction and efficient ways of obtaining preimages of G in [MP12].

2.4.6 Small secrets

All lattice based cryptographic constructions make an extensive use of *small* vectors. For this reason it is of paramount importance to know how to prove knowledge of some small vectors. Some problems may arise, one is efficiency and the other is a possible soundness gap, that is, knowing a vector with norm smaller than β the prover is only able to prove knowledge of a vector of size smaller than $\gamma\beta$, where γ may be a constant, polynomial or even exponential in the size of the vector in the worst case.

We can write the encryption of a message z in matrix form:

$$\begin{pmatrix} u_1 \\ \vdots \\ u_n \\ v_1 \\ \vdots \\ v_n \end{pmatrix} = \left\lfloor \frac{q}{2} \right\rceil \begin{pmatrix} 0 \\ \vdots \\ 0 \\ z_1 \\ \vdots \\ z_n \end{pmatrix} + \begin{pmatrix} a_1 & -a_n & -a_{n-1} & \cdots & -a_2 \\ a_2 & a_1 & -a_n & \cdots & -a_3 \\ a_3 & a_2 & a_1 & \cdots & -a_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_n & a_{n-1} & a_{n-2} & \cdots & a_1 \\ b_1 & -b_n & -b_{n-1} & \cdots & -b_2 \\ b_2 & b_1 & -b_n & \cdots & -b_3 \\ b_3 & b_2 & b_1 & \cdots & -b_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_n & b_{n-1} & b_{n-2} & \cdots & b_1 \end{pmatrix} \begin{pmatrix} r'_{\mathrm{E},1} \\ r'_{\mathrm{E},2} \\ r'_{\mathrm{E},3} \\ \vdots \\ r'_{\mathrm{E},n} \end{pmatrix} + \begin{pmatrix} e'_{u,1} \\ \vdots \\ e'_{u,n} \\ e'_{v,1} \\ \vdots \\ e'_{v,n} \end{pmatrix}$$

$$\begin{pmatrix} u_1 \\ \vdots \\ u_n \\ v_1 - \left\lfloor \frac{q}{2} \right\rceil z_1 \\ \vdots \\ v_n - \left\lfloor \frac{q}{2} \right\rceil z_n \end{pmatrix} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{Id}_n & \boldsymbol{0}_n \\ \boldsymbol{B} & \boldsymbol{0}_n & \boldsymbol{Id}_n \end{pmatrix} \begin{pmatrix} r'_{\mathrm{E},1} \\ \vdots \\ r'_{\mathrm{E},n} \\ e'_{u,1} \\ \vdots \\ e'_{u,n} \\ e'_{v,1} \\ \vdots \\ e'_{v,n} \end{pmatrix} \tag{6}$$

$$\boldsymbol{y} = \boldsymbol{A}'\boldsymbol{x} \quad \Big| \quad \|\boldsymbol{x}\|_\infty \leq \beta$$

The problem of proving knowledge of small random elements $r'_{\mathrm{E}}, e'_u, e'_v$ is equivalent to prove knowledge of a vector $\boldsymbol{x}$ such that $\boldsymbol{A}'\boldsymbol{x} = \boldsymbol{y} \mod q$ with $\|\boldsymbol{x}\|_\infty \leq \beta$, that is, we know a solution of the Inhomogeneous Short Integer Solution (ISIS) problem. There are two main approaches in the literature. We first consider the proposal by Ling *et al.* in [LNSW13], based on a code based construction of Stern [Ste96]. We have to consider some modifications, in our case the security of the proof is given by the hardness of solving the RLWE problem, instead of the general ISIS problem. Using RLWE we can take advantage of faster matrix vector multiplications as they can be seen as polynomial multiplications, for which FFT can be used. We expose here a brief description of the protocol of Ling *et al.*

Given $(\boldsymbol{A}, \boldsymbol{y})$ we want to prove knowledge of $\boldsymbol{x}$ such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}, \|\boldsymbol{x}\| \leq \beta$. This protocol, when knowing $\|\boldsymbol{x}\| \leq \beta$ allows us to prove $\|\boldsymbol{x}\| \leq 2\beta$. Carefully choosing β or slightly modifying the protocol it is possible to remove this gap.

The idea consists in proving that it is possible to write the coefficients of $\boldsymbol{x}$ with $k = \lceil \log \beta \rceil$ bits ($\{-1, 0, 1\}$). In order not to reveal those bits, elements from $\{-1, 0, 1\}$ are added so that the number of

each of them is the same, and then the result is permuted.

$$\begin{aligned}
&\boldsymbol{x} = (x_1, \dots, x_m) \\
&x_i = \sum_{j=0}^{k-1} 2^j b_{i,j}, \quad b_{i,j} \in \{-1, 0, 1\} \\
&\widetilde{\boldsymbol{u_j}} = (b_{1,j}, \dots, b_{m,j}), \quad \widetilde{\boldsymbol{u_j}} \in \{-1, 0, 1\}^m, \quad \boldsymbol{x} = \sum_{j=0}^{k-1} 2^j \widetilde{\boldsymbol{u_j}} \\
&\boldsymbol{u_j} = (\widetilde{\boldsymbol{u_j}} | \boldsymbol{t_j}) \qquad \text{(so that it has the same number of } -1, 0 \text{ and } 1) \\
&\boldsymbol{A'} = (\boldsymbol{A}|\boldsymbol{0}), \quad \boldsymbol{A'} \sum_{j=0}^{k-1} 2^j \boldsymbol{u_j} = \boldsymbol{y}
\end{aligned}$$

During the proof the prover randomly chooses:

$$\begin{aligned}
\boldsymbol{r_0}, \dots, \boldsymbol{r_{k-1}} &\xleftarrow{\$} \mathbb{Z}_q^{3m} \\
\pi_0, \dots, \pi_{k-1} &\xleftarrow{\$} \mathfrak{S}_{3m}
\end{aligned}$$

And generates the following commitments:

$$\begin{aligned}
c_1 &= \mathrm{Com}\left(\pi_0, \dots, \pi_{k-1}, \boldsymbol{A'} \sum_{j=0}^{k-1} 2^j \boldsymbol{r_j}\right) \\
c_2 &= \mathrm{Com}\left(\pi_0(\boldsymbol{r_0}), \dots, \pi_{k-1}(\boldsymbol{r_{k-1}})\right) \\
c_3 &= \mathrm{Com}\left(\pi_0(\boldsymbol{r_0} + \boldsymbol{u_0}), \dots, \pi_{k-1}(\boldsymbol{r_{k-1}} + \boldsymbol{u_{k-1}})\right)
\end{aligned}$$

The verifier randomly chooses a challenge in $\{1, 2, 3\}$ and the other two corresponding commitments are opened.

- If c_2, c_3 are revealed it is proved that $\boldsymbol{u_j}$ are indeed small.
- If c_1, c_3 are revealed it can be checked that the sum of $\boldsymbol{u_j}$ is $\boldsymbol{y}$ (if the element committed in c_1 is the sum of $\boldsymbol{r_j}$).
- If c_1, c_2 are revealed then it can be checked that there is no cheating on the $\boldsymbol{r_j}$, they are permuted as expected and their sum is the one that was compromised.

This protocol can be simulated $1/3$ of the times, therefore it is needed to repeat it several times to achieve a high soundness. Commitments to the permutations are done committing to the seed that would generate pseudorandom permutations.

Benoît *et al.* extended this idea based on the Stern protocol for small secrets that satisfy non-linear relations [LLM+16]. Xie *et al.* particularized the same protocol to ring lattices in [XXW13], allowing them to proof knowledge of a secret that satisfies any polynomial relation, with an almost optimal gap factor (or even no gap factor with the cost of larger communication complexity). However they still have the problem of only having a soundness of $2/3$ and make use of a specific commitment scheme.

Recently Cramer *et al.* proposed an amortized protocol [CDXY17], improved by del Pino and Lyubashevsky [dPL17], where they are able to reduce the size of the proofs when they are used to prove multiple

times the knowledge of different witnesses of the same linear relation. More improvements are being made for this amortized proofs, with new approaches as the one from Baum and Lyubashevsky proposing a novel protocol [BL17]. The one that best fits our requirements is [dPL17], as it achieves a constant gap, requires a small quantity of instances to achieve amortization and can be optimized for ring lattices.

This protocol is based on the rejection sampling method explained in theorem 2.24, combined with imperfect proofs.

- The prover creates many masking parameters and commits to them.
- The verifier asks the prover to reveal a fraction of the masking parameters to check that they are small. It also sends challenges.
- The prover answers as an standard proof of knowledge but aborts with a probability given by theorem 2.24.
- The verifier checks everything.

However, once we check the properties of this protocol we find that it is only an imperfect proof, that is, the verifier is convinced that the prover knows a small $\boldsymbol{x}$ such that $f(\boldsymbol{x}) = \boldsymbol{y}$ for all instances except for a fixed quantity k. There is a standard technique for transforming and imperfect proof of knowledge into a proof of knowledge, consisting in repeating the process several times with some instances constructed from the original ones. Even with this requirements this is the most efficient zero-knowledge proof we have found in the literature, as it allows us to perform many optimizations. The fraction of commitments that has to be revealed can be optimized depending on the number of instances and all the random elements can be calculated from seeds forming a Merkel Tree, largely decreasing the communication cost and the final size of the proof.

3. Voting process

Electronic voting might allow voters to vote from their personal computers or even their mobile phones. While this can be seen as an advantage that could be simpler than postal vote, some concerns may arise about its security. Some of them might even have to be considered when voting from a dedicated machine in a polling station. Once the voting option is encrypted it is signed with the private key of the voter, sent to the voting server and published in the bulletin board. Looking at the bulletin board the voter can check that his vote has been correctly casted, and verifying the following proofs he can be convinced that his vote has been counted in the final tally. But even though the vote has been counted the voter does not know if the encrypted vote contains his vote option or a different one. A corrupted computer could also reveal the vote to a third party.

In order to avoid the first malfunction it is not possible to reveal to the voter the random coins used in the encryption, as then he could prove his vote choice to a third party and sell his vote. We need a way to prove the voter that the encrypted vote stored in the bulletin board contains a specific vote choice, without allowing him to prove this to others, and without relaying on the honesty of his voting machine (maybe relaying on the honesty of different devices). That is, we need the coercion resistance cast as intended property. A malicious machine should not be able to modify the option of an honest voter, a corrupted voter using an honest machine should not be able to prove his vote to a corruptor.

3.1 Coercion resistant cast as intended

One of the options presented in the literature are return codes. We want to avoid the possibility of our computer cheating on us, so we made the voting server to prove us that it has received the option we wanted. We do not want it to know what is our option, so the process takes several steps.

- During the registration in the election the voter applies a secret deterministic one way function to each of the voting options, the images are called partial return codes.
- Also during the registration the voting server applies another secret deterministic one way function to the partial return codes, and prints them in a card that is given to the voter (or is handled to him through a secure channel).
- During the voting the personal computer sends to the voting server the signed encrypted voting option, the corresponding partial return code and a proof of this partial return code being the one corresponding to the encrypted option.
- The voting server receives the signed encrypted vote and the partial return code with its proof, checks the proof, applies its secret deterministic one way function to the partial return code and sends it back to the voter.
- The voter checks in the card that the return code corresponds with the option he has chosen.

Those secret functions should behave properly. They should be difficult to invert even if there is a small set of possible preimages, and also difficult to compute one preimage even if we know the image of another element.

This strategy is coercion resistant, only the voter and the server known the return codes, so if the voter receives a return code he is convinced that the return code has been computed by the server. However, he

can not convince a third party, as this third party does not known that the return code is an answer from the server or just the voter showing him the corresponding return code from his card.

The main disadvantage of this approach, besides the requirement of different devices that should not collude, one for voting and one for receiving the return codes, is the active participation of the voter. Only the voter is able to verify that the return code is what was expected to be, but we do not know if he has carefully checked it. We aim for a universally verifiable coercion resistance cast as intended, that is, if the voter is honest anyone should be able to verify that his encrypted vote corresponds to his option, without revealing it.

Another simpler option is known as *Challenge or cast*. Once the vote is encrypted the voter sees it and can decide to cast it or to reveal the random coins to verify that it encrypts his option, and then generate another ciphertext. The voter can order the computer to reveal the random coins for the ciphertexts as many times as he wants, verifying that the machine is not cheating. A corrupted computer would need to encrypt his option m for all ciphertexts until it predicts that the voter is going to decide to cast his vote, and then encrypt m'. Even if this might be mathematically satisfactory it is less convincing for the voter. The main argument against challenge or cast is that randomness should be consider a resource, and is a bad idea to rely on the voter as a random source. Besides being theoretically secure a scheme needs to seem secure, and it might be less convincing for the voter to verify all the encryptions except the one that really counts.

Other proposals try to build a challenge and cast scheme. A naive approximation would be an Interactive Zero-Knowledge Proof of Knowledge (IZKPoK). The voter is convinced as he has been interacting with his computer, but the proof would mean nothing to a third party, as it could be simulated. However this does not prevent coercion, as a corruptor could force the voter to answer with a hash of the commitment, making the IZKPoK into a NIZKPoK, and then handling it to him, proving his voting option.

3.2 Our proposal for a coercion resistant cast as intended proof

We present here how to build a coercion resistant cast as intended proof, following the general proposal given by Guasch in [GC16].

We specifically want an or-proof of two things, either a ciphertext contains a given message or the proof has been constructed by the voter.

$$\text{ZK-proof}\left[\, w \,\middle|\, (w \text{ witness of } (u,v) \text{ being an encryption of } m) \vee (w \text{ witness of } \mathcal{P} \text{ being } \texttt{id}) \,\right]$$

That is, it proves that either a given ciphertext encrypts a particular message or the prover is a particular entity `id`, that will be the voter. If his personal computer produces a proof like this the voter knows that the computer is not him, and is convinced that the ciphertext is an encryption of his selected option. However if the voter handles this proof to a third party it would be useless, as it does not prove anything relevant to the corrupter.

In practice this is achieved with a special kind of proof that can be simulated knowing a secret key. This secret key is given to the voter but is not introduced into his personal computer. When the computer makes the proof for option m the prover is convinced that his vote (u, v) encrypts m. Then he can introduce the secret key and generate a fake proof of (u, v) being an encryption of m', to show to any possible corruptor.

To have this kind of property we have to introduce the concept of chameleon hashes:

Definition 3.1 (Chameleon Hash [KR97])**.** CHAM-HASH

A *chameleon hash function* is associated to a user id who has published a hashing public key $HK_{\mathtt{id}}$ and holds the corresponding secret key $CK_{\mathtt{id}}$. The public $HK_{\mathtt{id}}$ defines a function $\text{CHAM-HASH}_{\mathtt{id}}(\cdot,\cdot)$, that can be efficiently computed knowing $HK_{\mathtt{id}}$. Taking as input a message m and a random element r this function generates a hash value $\text{CHAM-HASH}_{\mathtt{id}}(m, r)$ satisfying the following properties:

- **Collision resistance**: there exists no PPT algorithm on input $HK_{\mathtt{id}}$ able to find pairs (m_1, r_1), (m_2, r_2) with $m_1 \neq m_2$ such that $\text{CHAM-HASH}_{\mathtt{id}}(m_1, r_1) = \text{CHAM-HASH}_{\mathtt{id}}(m_2, r_2)$, except with a negligible probability.
- **Trapdoor collisions**: there exists an efficient algorithm that takes as input the secret key $CK_{\mathtt{id}}$, any pair (m_1, r_1) and an additional message m_2, and finds an element r_2 such that $\text{CHAM-HASH}_{\mathtt{id}}(m_1, r_1) = \text{CHAM-HASH}_{\mathtt{id}}(m_2, r_2)$.

Krawczyk and Rabin show in [KR97] a generic construction for chameleon hash functions, using a particular pair of trapdoor permutations, and more efficient constructions based on the hardness of factoring and the discrete logarithm.

We propose to use a lattice based hash function [GGH11] implemented with trapdoor functions and prove that it is a chameleon hash function whose security relies on lattice based assumptions believed to be quantum secure.

Let $\boldsymbol{A}_1$ be a random matrix and $\boldsymbol{A}_2$ a matrix obtained as in section 2.4.5, for which the SIS problem can be efficiently solvable with the secret key. D and R are sets of *small* vectors of the corresponding size. Consider the following function:

$$\boldsymbol{A} = (\boldsymbol{A}_1|\boldsymbol{A}_2) \in \mathbb{Z}_q^{n\times m}, \boldsymbol{A}_1 \in \mathbb{Z}_q^{n\times w}, \boldsymbol{A}_2 \in \mathbb{Z}_q^{n\times(m-w)}, \boldsymbol{x} \in \mathbb{Z}_q^{w}, \boldsymbol{r} \in \mathbb{Z}_q^{m-w}$$

$$\begin{aligned} f_{\boldsymbol{A}}\colon D \times R \subset \mathbb{Z}_q^w \times \mathbb{Z}_q^{m-w} &\to \mathbb{Z}_q^n \\ (\boldsymbol{x}, \boldsymbol{r}) &\mapsto \boldsymbol{A}(x^\intercal|r^\intercal)^\intercal \end{aligned}$$

$$f_{\boldsymbol{A}}(\boldsymbol{x}, \boldsymbol{r}) = \begin{pmatrix}\boldsymbol{A}_1\boldsymbol{A}_2\end{pmatrix}\begin{pmatrix}\boldsymbol{x}\\ \boldsymbol{r}\end{pmatrix} = \boldsymbol{A}_1\boldsymbol{x} + \boldsymbol{A}_2\boldsymbol{r} = \boldsymbol{y}$$

Theorem 3.2. *The above function is a chameleon hash function if $\boldsymbol{r}$ is chosen from R following a discrete gaussian distribution and the SIS problem can not be efficiently solved.*

Chameleon Hash. Let $\boldsymbol{A} = (\boldsymbol{A}_1|\boldsymbol{A}_2)$.

- **Collision resistance**:

$$\begin{aligned} \boldsymbol{x}, \boldsymbol{r}, \boldsymbol{x}', \boldsymbol{r}' \leftarrow \mathcal{A} \text{ s.t. } & f_{\boldsymbol{A}}(\boldsymbol{x}, \boldsymbol{r}) = f_{\boldsymbol{A}}(\boldsymbol{x}', \boldsymbol{r}'), \boldsymbol{x}, \boldsymbol{x}' \in D, \boldsymbol{r}, \boldsymbol{r}' \in R, \\ \implies & \boldsymbol{A}(\boldsymbol{x}^\intercal|\boldsymbol{r}^\intercal)^\intercal = \boldsymbol{A}(\boldsymbol{x}'^\intercal|\boldsymbol{r}'^\intercal)^\intercal \\ \implies & \boldsymbol{A}(\boldsymbol{x}^\intercal - \boldsymbol{x}'^\intercal|\boldsymbol{r}^\intercal - \boldsymbol{r}'^\intercal)^\intercal = 0 \qquad \text{(solve SIS)} \end{aligned}$$

- **Trapdoor collisions**: Given $\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{x'}$ using CK we find $\boldsymbol{r'} \in R$ such that:

$$\begin{aligned} \boldsymbol{A}_2\boldsymbol{r'} &= \boldsymbol{A}_1\boldsymbol{x} + \boldsymbol{A}_2\boldsymbol{r} - \boldsymbol{A}_1\boldsymbol{x'} \\ \implies f_{\boldsymbol{A}}(\boldsymbol{x}, \boldsymbol{r}) &= f_{\boldsymbol{A}}(\boldsymbol{x'}, \boldsymbol{r'}) \end{aligned}$$

$\boldsymbol{A}_2 \in \mathbb{Z}_q^{n\times(m-w)}$ is at a negligible statistical distance of a uniform distribution if $(m-w) \approx 2n\log q$ [MP12]. This can be reduced even more if we only ask computational indistinguishability, getting a worse backdoor quality. □

In order to prove that a ciphertext contains a vote for a particular candidate z the machine has to prove knowledge of a solution to the ISIS problem described by eq. (6). In order to do this it uses the proof proposed by Ling *et al.* that we have seen in section 2.4.6. To achieve enough soundness it has to be repeated t times, so it computes t commitments and stores them in a vector $\boldsymbol{c}$.

Then we use as a hash the following function $H_2(\text{CHAM-HASH}_A(H_1(\boldsymbol{c}), \boldsymbol{r}))$. H_1 is a hash function that sends t-tuples of commitments to small vectors in $\mathbb{Z}_q^w$, $\boldsymbol{r}$ is a random *small* vector chosen from a discrete gaussian distribution, CHAM-HASH_A is a chameleon hash function from $\mathbb{Z}_q^m$ to $\mathbb{Z}_q^n$ given by $\text{CHAM-HASH}_A(\boldsymbol{x}, \boldsymbol{r}) = (\boldsymbol{A}_1|\boldsymbol{A}_2)\begin{pmatrix}\boldsymbol{x}\\ \boldsymbol{r}\end{pmatrix} = \boldsymbol{A}_1\boldsymbol{x} + \boldsymbol{A}_2\boldsymbol{r} = \boldsymbol{y}$, and H_2 is another hash function that sends vectors from $\mathbb{Z}_q^n$ to the space of challenges $d \in \{1, 2, 3\}^t$.

Using those challenges it computes the answers e and shows the transcription with the random element $\boldsymbol{r}$ to the voter. The voter is convinced if the transcription is valid and the challenges are indeed this chain of hashes using the random small element $\boldsymbol{r}$.

He can simulate any proof choosing at random a vector $h \xleftarrow{\$} \mathbb{Z}_q^n$. Then applying the second hash obtains $d' = H_2(h)$ and uses it to simulate t proofs (c', d', e'). As he knows the chameleon hash backdoor he can compute $\boldsymbol{r'}$ so that h is a chameleon hash of $H_1(c')$. This way $d' = H_2(h) = H_2((\text{CHAM-HASH}_A(H_1(c'), \boldsymbol{r'}))$ and the proof will be valid.

Observation 3.3. In the end the voter should have access to the secret trapdoor, for which he would need another device, that could be a mobile phone, a cryptographic card, another computer, etc. Security comes from both devices not being corrupted at the same time, as only a coalition of corrupted devices could produce all fake proofs.

This scheme can be slightly adapted to make it universally verifiable, that is, not only the voter but anyone is able to verify the coercion resistant cast as intended proof. If there is a limited number of candidates then each voter could have a different hash functions for each of the candidates. During the voting process the voter reveals to its computer the secret hash keys for all the candidates except the one he has voted. Then, the computer is able to generate proofs of the ciphertext being an encryption of all the possible voting options (all but one will be simulated) and publishes them on the bulletin board. No information is revealed as all proofs are indistinguishable and everybody can check that, if the voter has not revealed the secret information for his option, then the vote has been casted as intended.

In order to reduce the communication cost a Merkle tree structure can be used. Each user only needs to keep a unique 256 bits seed s. It can be split into two parts of 128 bits, and then with an extendable output function (XOF) transform each part into a 256 bits seed again, obtaining s_0 and s_1. This process

can be repeated as many times as needed, following a tree structure until we have as many leaves as voting options. From each of those seeds on the leaves an extendable output function can be used to generate the secret key of the chameleon hash function.

In fig. 1 example we assume 8 voting options labeled in binary from 000 to 111 and suppose we want to vote for option 011. Then, to communicate all secret keys except the one corresponding to 011 we only need to send s_{00}, s_{010} and s_1, from which we can compute all the missing leaves except s_{011}. With this the communication cost is logarithmic in the size of voting options, that is a small number in most of the cases.

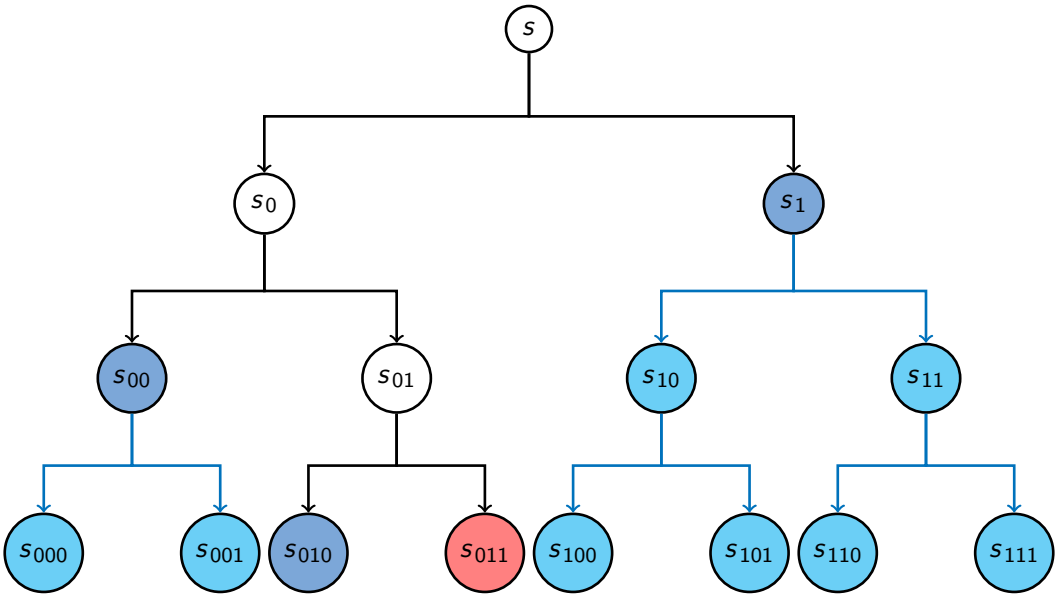


Figure 1: Merkle tree example

4. Mixing process

In an analogy with a traditional election, encrypting the votes is equivalent to introducing the ballots into envelopes, as this guarantees that nobody can see what is inside until they are opened (decrypted). Despite that, all envelopes are equal and indistinguishable, and it is easy to shuffle votes. In an electronic election encrypted votes are unique, linked to the person that casted them, and they can not be directly decrypted.

There are two main alternatives to deal with this problem. If the election is simple enough (the final result is the sum of all the votes) one option for tally voting consists on using a homomorphic encryption scheme and adding up all encryptions. Then, only the final result is decrypted.

However this does not work with the encryption scheme we have selected, even if it can be adapted to encode messages in r-ary instead of binary multiplying the message by $\lfloor \frac{q}{r} \rceil$ instead of $\lfloor \frac{q}{2} \rceil$. If there are N voters, to avoid decryption error the accumulated error terms should have norm smaller than $\lfloor \frac{q}{2N} \rceil$, but, if ciphertexts are directly summed up the errors can grow up to a factor N. This means we should have a quadratic margin on the number of voters, that would imply a larger q and hence larger ciphertexts. Another inconvenient is that it requires that options can be encoded as something that can be summed to obtain the final result. This is not possible for complex election systems where the result is obtained from the votes and some rules, for example if voters can delegate their votes, choose different preferences or any other possibility. Usually we also encode the votes with some error correcting code before encrypting them, so that we can minimize even more the possibility of a decoding error, and this code might not be homomorphic.

The other alternative are mixing networks (mix-nets), a much more flexible approach. A mix-net consists of several nodes that permute and re-encrypt the ciphertexts. If at least one mix-node is honest and does not reveal its permutation then re-encryption means that it is not possible to relate its output to its input, making impossible to relate the votes that are finally decrypted with the ones that were originally received by the voting server.

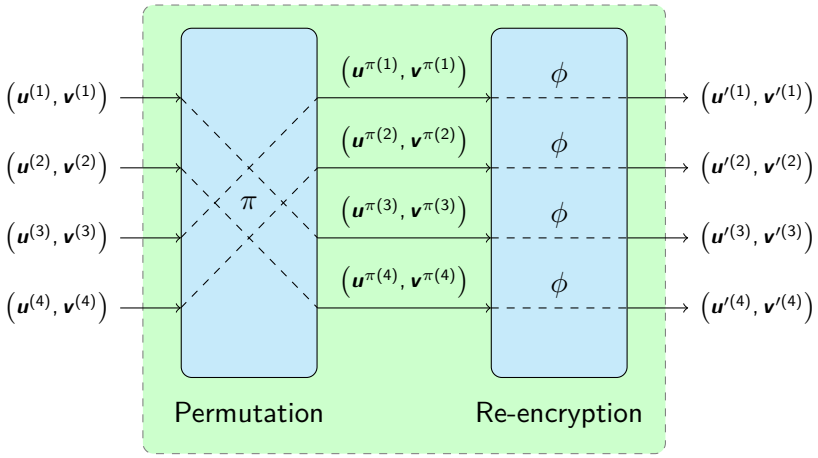


Figure 2: Mixing node

To ensure that no vote has been modified, removed or inserted, all inputs and outputs are published on the bulletin board and each node has to proof that the output is indeed a permutation and a re-encryption of its input. This particular operation of permuting and re-encrypting is called a shuffle.

Observation 4.1. What we call a proof of a shuffle only ensures that the ciphertexts from the output

encrypt the same messages as the ciphertexts from the input, but does not guarantee that the random re-encrypting parameters are chosen from the prescribed distributions. A malicious node could select some pathological re-encrypting parameters that allow adversaries to know the relation between the input and the output, and still construct the proof. A proof of a shuffle does not prove privacy, it only proofs that the node has not modify its content. The security is only achieved if at least one node correctly follows the re-encryption algorithm and chooses its parameters from the adequate truly random distributions.

This is the expected requirement of this proof. Proving that re re-encryption has been computed using random elements from the required distribution is harder and adds no extra security, as a malicious mix-node could just leak the permutation.

4.1 Related work

Mix-nets were first introduced by Chaum in [Cha81]. His idea was to encrypt the messages as many times as mixing nodes using RSA onions with random padding. Each node would decrypt the outer layer and remove the padding, so that the output of the final node would be the plaintexts. This proposal required a ciphertext size proportional to the number of nodes, as was pointed out by Park *et al.* [PIK94], who proposed a re-encryption mix-net, where ciphertexts are re-randomized, and only decrypted at the end. They also proposed a version where nodes partially decrypt the ciphertexts and re-randomized them. Sako and Kilian give the first definition of universally verifiable mix-net, providing a zero-knowledge proof of correct shuffling that can be verified by everybody [SK95]. Many proposals and improvements for efficiently verifiable mix-nets have been presented since then, due to the work of Abe *et al.* [Abe98, Mas99, AH01], Furuwaka *et al.* [FS01, Fur05] and Wikström *et al.* [Wik04, Wik05, Wik09, TW10]. Neff *et al.* proposed a verifiable shuffle in [Nef01] that was improved in [Nef03] by Neff himself and in [Gro03, GI08, BG12] by Groth *et al.* Our proposal is based on this last proposal [BG12] of Bayer and Groth.

To the best of our knowledge there are very few proposals for lattice based re-encryption mix-nets. Singh *et al.* [SRB14, SRB15] propose a re-encryption mix-net for lattice based cryptography, but they do not provide any proof of shuffling. Costa *et al.* [CMM17] proposed the first proof of a shuffle for lattice based encryption schemes, based on the proof of Terelius and Wikström [TW10]. However they use Pedersen commitments, which are perfectly hiding but only computationally binding under the discrete logarithm assumption. This implies that proofs using this protocol will not leak any information in a future with quantum computers, and are long term secure, but will not longer be usable in this future as soundness is based on the discrete logarithm assumption. We aim to present a universally verifiable proof of a shuffle that is entirely based on post-quantum assumptions.

4.1.1 Costa, Martínez and Morillo proof of a shuffle

We first summarize here the proposal of Costa, Martínez and Morillo [CMM17], and explain why it can not be efficiently adapted to fully post-quantum constructions.

It follows Wikström's idea of characterizing a permutation with its permutation matrix, commit to the matrix and then prove in zero knowledge that the commitment is indeed a commitment to a permutation matrix and that the output of the node is a re-randomization of the input multiplied by this matrix. The matrix is committed by columns using Generalized Pedersen Commitments, which allows the prover to commit to an $N \times N$ matrix using N commitments, achieving a linear cost. In our case, if we commit separately to each of the elements with Benhamouda *et al.* commitment scheme it would imply to use N^2 commitments. Taking into account that Benhamouda *et al.* commitments are vectors of polynomials the

size would be considerably large for a $\{0,1\}$-matrix. Trying to commit to tuples of bits as coefficients of polynomials would seem a solution, but it does not work, as then it is not possible to relate commitments to the tuple with commitments to each of its coefficients, due to divisors of zero that make some operations non invertible ($\mathbb{Z}_q[x]/\langle x^n+1\rangle$ is a ring while $\mathbb{Z}_q$ was a field). This same problem arises in some of the required equations needed to proof soundness on the protocol. For this reason a different strategy must be followed if we want to achieve post-quantum soundness and zero-knowledge.

4.1.2 Bayer and Groth proof of a shuffle

Bayer and Groth proposed an efficient proof of a shuffle based on a completely different approach. Instead of characterizing a permutation by its permutation matrix they focus on the elements permuted, and use that sets $A = \{a_i\}_{i\in 1,\dots,N}$ and $B = \{b_i\}_{i\in 1,\dots,N}$ are equal if the polynomials that have them as roots are equal:

$$\prod_{i=1}^{N}(x-a_i) = \prod_{i=1}^{N}(x-b_i)$$

Then, there exists a permutation π such that $a_i = b_{\pi^{-1}(i)}$.

It is also useful to write them as coefficients of the polynomials:

$$\sum_{i=1}^{N} a_i x^i = \sum_{i=1}^{N} b_i x^{\pi(i)}$$

To verify that two polynomials are equal one possibility is to evaluate them in a random point and see if their evaluations are equal. To take into account and bound the probability of two different polynomials having the same evaluation on a random point we use the Schwartz-Zippel lemma.

Lemma 4.2 (Schwartz-Zippel lemma)**.** *Let p be a non-zero multivariate polynomial of degree d over $\mathbb{Z}_q$, then the probability of $p(x_1,\dots,x_n)=0$, for randomly chosen $x_1,\dots,x_n \xleftarrow{\$} \mathbb{Z}_q^*$ is at most $\frac{d}{q-1}$.*

Several difficulties arise. The permutation π has to remain secret, so the prover can commit to permuted elements, but can not open those commitments directly. Another problem comes from the random evaluation points, as they have to be chosen by the verifier after the polynomials are fixed (otherwise we would not be able to apply Schwartz-Zippel lemma). However $\mathcal{P}$ is not able to commit to an abstract algebraic element like a polynomial $\sum_{i=1}^{N} b_i x^{\pi(i)}$ and then transform it into a commitment to one evaluation of it $\sum_{i=1}^{N} b_i \alpha^{\pi(i)}$ in a random point α chosen by $\mathcal{V}$. To elude this issue the prover commits to π, sends the commitment to the verifier and proves knowledge of an opening. Then the verifier chooses the random element α and the prover commits to powers of it $\left\{\alpha^{\pi(i)}\right\}_{i=1}^{N}$ in $c_{\alpha^{\pi(i)}}$ (indexed by i) permuted in the order previously fixed by π. At that point $\mathcal{P}$ has to prove that the commitment to π was an actual commitment to a permutation, and that it was used to honestly build the commitments to $\left\{\alpha^{\pi(i)}\right\}_{i=1}^{N}$. Therefore, when the prover uses those commitments to construct a final commitment to $\sum_{i=1}^{N} b_i \alpha^{\pi(i)}$ it is clear that this construction is only defined by α and the permutation π, and the later was already fixed before α was chosen. The prover can finally open this last commitment to $\sum_{i=1}^{N} a_i \alpha^i$, and prove that elements of A and B are the same, in a different order.

The exact details involve more polynomials and evaluation points, and are given in section 4.2, with some modifications, as we have adapted this part of the proof for our protocol.

The output of each mix-net node is not only a permutation of the input, but also a re-encryption, thus we need to introduce it on the proof. Bayer and Groth use ElGamal encryptions. They re-encrypt their ciphertexts multiplying by an encryption of 1. The homomorphic properties of ElGamal imply that when we combine all encryptions in the polynomial in α we can also combine all those re-encrypting parameters into another encryption of the multiplication of all of those 1's, which is again a 1, with the corresponding randomization elements given by α. To combine it with the previous proof it uses a multi-exponentiation and a product argument.

We use the fact that in our RLWE encryption scheme adding an encryption of 0 works as a re-encryption, but it is only true if the number of re-encryptions is limited. If we consider an arbitrary linear combination of the randomization elements the result would not have small norm, and it would not be a valid encryption. Therefore, we have substituted this part of their argument and treated all randomization elements separately.

4.2 Our proposal for a proof of a shuffle

In this section we are going to explain how to construct a cryptographic shuffle of a list of N RLWE encryptions and also how to prove its correctness with a post-quantum cryptosystem.

When we permute and then re-randomize a vote we will use the following notation $\left(u'^{(i)}, v'^{(i)}\right) = \text{Re-Enc}\left(\left(u^{\pi(i)}, v^{\pi(i)}\right), r_{\mathsf{E}}'^{(i)}, e_{\mathsf{E},u}'^{(i)}, e_{\mathsf{E},v}'^{(i)}\right)$. Following this notation with a given permutation π and a set of re-encryption parameters $\left(r_{\mathsf{E}}'^{(i)}, e_{\mathsf{E},u}'^{(i)}, e_{\mathsf{E},v}'^{(i)}\right)$ for each one of the messages, we can express the shuffling of N RLWE encryptions as:

$$\begin{pmatrix} u'^{(1)}, & v'^{(1)} \\ \vdots & \vdots \\ u'^{(N)}, & v'^{(N)} \end{pmatrix} = \begin{pmatrix} u^{\pi(1)}, & v^{\pi(1)} \\ \vdots & \vdots \\ u^{\pi(N)}, & v^{\pi(N)} \end{pmatrix} + \begin{pmatrix} r_{\mathsf{E}}'^{(1)} \\ \vdots \\ r_{\mathsf{E}}'^{(N)} \end{pmatrix} \begin{pmatrix} a_{\mathsf{E}}, & b_{\mathsf{E}} \end{pmatrix} + \begin{pmatrix} e_{\mathsf{E},u}'^{(1)}, & e_{\mathsf{E},v}'^{(1)} \\ \vdots & \vdots \\ e_{\mathsf{E},u}'^{(N)}, & e_{\mathsf{E},v}'^{(N)} \end{pmatrix} \tag{7}$$

A mix-net node should prove that it knows the permutation π and the random elements $r_{\mathsf{E}}'^{(i)}, e_{\mathsf{E},u}'^{(i)}, e_{\mathsf{E},v}'^{(i)}$ such that the output of the node $\begin{pmatrix} \boldsymbol{u'} & \boldsymbol{v'} \end{pmatrix}$ is the input $\begin{pmatrix} \boldsymbol{u} & \boldsymbol{v} \end{pmatrix}$ re-encrypted and permuted, without revealing any information about π, $r_{\mathsf{E}}'^{(i)}$, $e_{\mathsf{E},u}'^{(i)}$ and $e_{\mathsf{E},v}'^{(i)}$.

$$\text{ZK-proof}\left[\begin{array}{c|c} \begin{matrix} \pi \\ r_{\mathsf{E}}'^{(1)}, \dots, r_{\mathsf{E}}'^{(N)} \\ e_{\mathsf{E},u}'^{(1)}, \dots, e_{\mathsf{E},u}'^{(N)} \\ e_{\mathsf{E},v}'^{(1)}, \dots, e_{\mathsf{E},v}'^{(N)} \end{matrix} & \begin{matrix} \left(u'^{(1)}, v'^{(1)}, \dots, u'^{(N)}, v'^{(N)}\right)^{\mathsf{T}} \\ = \\ \begin{pmatrix} \text{Re-Enc}\left(\left(u^{\pi(1)}, v^{\pi(1)}\right), r_{\mathsf{E}}'^{(1)}, e_{\mathsf{E},u}'^{(1)}, e_{\mathsf{E},v}'^{(1)}\right)^{\mathsf{T}} \\ \dots \\ \text{Re-Enc}\left(\left(u^{\pi(N)}, v^{\pi(N)}\right), r_{\mathsf{E}}'^{(N)}, e_{\mathsf{E},u}'^{(N)}, e_{\mathsf{E},v}'^{(N)}\right)^{\mathsf{T}} \end{pmatrix} \\ \left\|r_{\mathsf{E}}'^{(i)}\right\|, \left\|e_{\mathsf{E},u}'^{(i)}\right\|, \left\|e_{\mathsf{E},v}'^{(i)}\right\| \leq \delta \end{matrix} \end{array}\right]$$

We recall here that the set of mix-net nodes needs to achieve two properties, anonymity and integrity of the votes. On the one hand if at least one of the mix-net nodes chooses its re-encryption parameters

from the appropriate discrete gaussian distribution χ_σ^n and keeps them secret then it is not feasible for an adversary to trace the votes back, because it will mean that it will be able to distinguish RLWE samples from uniformly random polynomials.

On the other hand it needs to guarantee that the plaintexts of all the votes remain the same, even if some of the nodes are dishonest and leak the permutation they are not able to modify the encrypted plaintext. In order to do so it publishes the corresponding proofs of permutation and re-encryption. The mix-net node should convince everybody that the votes have been permuted and then there has been added to them something that is of the form of an encryption of 0. $(u_0, v_0) = (a_E r'_E + e'_{E,u}, b_E r'_E + e'_{E,v})$ with small polynomials $r'_E, e'_{E,u}, e'_{E,v}$ to certify that when we decrypt it we would get a 0. The way we have of achieving this property is to prove that the infinity norm of $r'_E, e'_{E,u}, e'_{E,v}$ is bounded by some parameter $\delta << q/4$.

This implies that honest nodes must choose its parameters form χ_σ^n conditioned to a norm smaller than δ. As it is explained in [BKLP15] for a suitable δ even if this additional restriction on the re-encryption parameters norm is applied, the RLWE re-encryptions remain pseudorandom, as the two probability distributions are statistically close.

Following the strategy used by Bayer and Groth in [BG12] we prove that two sets contain the same elements committing to two polynomials, each of them having as roots the elements from each set, and then prove that both polynomials are equal. To prove with overwhelming probability that two polynomials are equal it evaluates them and uses the Schwartz-Zippel lemma.

In our case the elements will be polynomials. We want to prove:

$$\left\{\left(u'^{(i)}, v'^{(i)}\right) - \left(a r_E'^{(i)} + e_{E,u}'^{(i)}, b r_E'^{(i)} + e_{E,v}'^{(i)}\right)\right\}_{i=1}^{N} = \left\{\left(u^{(i)}, v^{(i)}\right)\right\}_{i=1}^{N} \subset R_q \times R_q \tag{8}$$

Our polynomials have coefficients in R_q, that is, we will work in $R_q[A]$. If we have an equality as polynomials:

$$\sum_{i=1}^{N} A^i u^{(i)} = \sum_{i=1}^{N} A^{\pi(i)} \left(u'^{(i)} - a r_E'^{(i)} + e_{E,u}'^{(i)}\right)$$
$$\sum_{i=1}^{N} A^i v^{(i)} = \sum_{i=1}^{N} A^{\pi(i)} \left(v'^{(i)} - b r_E'^{(i)} + e_{E,v}'^{(i)}\right)$$

Then we should also have an equality when evaluating in a randomly chosen $\alpha \in R_q$:

$$\sum_{i=1}^{N} \alpha^i u^{(i)} = \sum_{i=1}^{N} \alpha^{\pi(i)} \left(u'^{(i)} - a r_E'^{(i)} + e_{E,u}'^{(i)}\right) \tag{9}$$
$$\sum_{i=1}^{N} \alpha^i v^{(i)} = \sum_{i=1}^{N} \alpha^{\pi(i)} \left(v'^{(i)} - b r_E'^{(i)} + e_{E,v}'^{(i)}\right) \tag{10}$$

For this to imply that over the choice of α the coefficients have overwhelming probability of being equal we will need a generalized version of the Schwartz-Zippel lemma that works in general commutative rings as R_q, that are not necessarily integral domains. This is lemma 4.3. It is also important to notice that all this equalities will be shown commited. As was first introduced in section 4.1.2 we will need to commit to

$\alpha^{\pi(i)}$ for a permutation π fixed before the verifier chooses the element α (for the lemma to be useful we need the polynomials in $R_q[A]$ to be determined before the evaluation point is fixed).

Lemma 4.3 (Generalized Schwartz-Zippel lemma)**.** *Let $P \in R[x_1, x_2, \dots, x_n]$ be a non-zero polynomial of total degree $d \geq 0$ over a commutative ring R. Let S be a finite subset of R such that:*

$$\forall s, t \in S : ((\exists u \in R : (u \neq 0 \wedge su = tu)) \implies s = t)$$

and let $r_1, r_2, \dots, r_n$ be selected at random independently and uniformly from S.

Then $Pr[P(r_1, r_2, \dots, r_n) = 0] \leq \frac{d}{|S|}$.

Proof. The condition imposed on S implies that a nonzero degree d univariate polynomial $f \in R[x]$ can only have d roots in S. We can prove this by induction.

Case $d = 0$ is trivially true.

Assume the inequality holds for polynomials of degree smaller or equal to d and let $f(x)$ be a polynomial of degree $d + 1$ with $d + 2$ different roots $a_1, a_2, \dots, a_{d+2} \in S$. The polynomial reminder theorem implies that we can write $f(x) = (x - a_{d+2})g(x)$ for some polynomial $g(x)$ of degree d.

In a field, $a_1, a_2, \dots, a_{d+1}$ being roots of $f(x)$ and not of $(x - a_{d+2})$ would imply that they are roots of $g(x)$. But we are working with a polynomial ring, that may not be an integral domain, and this may not always be true.

However as $a_1, \dots, a_{d+1}$ belong to S we can prove by contradiction that they are all roots of $g(x)$. Assume that $(a_i - a_{d+2})g(a_i) = 0$ with $g(a_i) \neq 0$, we have the condition of the lemma hypothesis. $g(a_i) \neq 0 \wedge a_i g(a_i) = a_{d+2} g(a_i)$. As both $a_i, a_{d+2} \in S$ we have that $a_i = a_{d+2}$, contradicting the hypothesis of the roots being different.

Therefore we have that $g(a_i) = 0$, and this is valid for all $i \in 1, \dots, d + 1$. Then $g(x)$ would have $d + 1$ different roots in S, this time contradicting the induction hypothesis and proving the result for the univariate case.

In order to prove the multivariate case we can follow the standard proof of the Schwartz–Zippel lemma, by induction on n.

Case $n = 1$ is the univariate case that we have already proved.

Assume the lemma is true for polynomials of n or less variables. We can write an $(n + 1)$-variate polynomial as:

$$f(x_1, \dots, x_{n+1}) = \sum_{i \leq d'} x_{n+1}^i f_i(x_1, \dots, x_n)$$

Where $f_{d'}$ is nonzero. As an n-variate polynomial, by the induction hypothesis, we have:

$$\Pr\Big[f_{d'}(x_1, \dots, x_n) = 0\Big] \leq \frac{\deg(f_{d'})}{|S|}$$

If $f_{d'}(a_1, \dots, a_n) \neq 0$, by the base case of the induction hypothesis we have:

$$\Pr\Big[f(a_1, \dots, a_n, x_{n+1}) = 0\Big] \leq \frac{d'}{|S|}$$

And finally:

$$
\begin{aligned}
\Pr\Big[f(a_1,\dots,a_n,a_{n+1})=0\Big] &= \Pr\Big[f(a_1,\dots,a_n,a_{n+1})=0 \wedge f_{d'}(a_1,\dots,a_n)=0\Big] \\
&\quad + \Pr\Big[f(a_1,\dots,a_n,a_{n+1})=0 \wedge f_{d'}(a_1,\dots,a_n)\neq 0\Big] \\
&\leq \Pr\Big[f_{d'}(a_1,\dots,a_n)=0\Big] \\
&\quad + \Pr\Big[f(a_1,\dots,a_n,a_{n+1})=0 \,\Big|\, f_{d'}(a_1,\dots,a_n)\neq 0\Big] \\
&\leq \frac{\deg(f_{d'})}{|S|} + \frac{d'}{|S|} \\
&\leq \frac{d}{|S|}
\end{aligned}
$$

□

Now we need to define a suitable subset $S \subseteq \mathbb{Z}_q[x]/\langle x^n+1\rangle$ for which the condition holds. If we rewrite:

$$\text{if there is an } u \neq 0 \text{ such that } (s-t)u = 0 \text{ it must be because } s = t$$

in terms of its contrapositive we get:

$$\text{the difference } s-t \text{ of any two different elements in } S \text{ is not a divisor of } 0$$

We can guarantee it if all differences of elements in S are invertible. We choose:

$$S = \Big\{ p(x) \in \mathbb{Z}_q[x]/\langle x^n+1\rangle \,\Big|\, \deg p(x) < n/2 \Big\}$$

Observe that the proposed subset S meets the required condition for lemma 4.3, as all differences of two polynomials in S have degree smaller than $n/2$, and $q \equiv 3 \mod 8$ required for section 2.4.4 implies that they are invertible. The number of elements in S is still exponential in n, so we can use it as a set of challenges.

Once that we have solved the problem of working on a ring instead of a field we can start building a proof for eqs. (9) and (10). The prover $\mathcal{P}$ starts choosing a random permutation $\pi \xleftarrow{\$} \mathfrak{S}_N$ and following Bayer and Groth he starts building commitments $\boldsymbol{c}_{\pi(1)},\dots,\boldsymbol{c}_{\pi(N)}$ ($\boldsymbol{c}_{\pi(i)}$ is indexed by i and not by the value of $\pi(i)$) to $\pi(1),\dots,\pi(N)$, using the commitment scheme described in section 2.4.4.

The prover sends those commitments to the verifier and he replies with a random polynomial $\alpha \xleftarrow{\$} S$. $\mathcal{P}$ commits to each power $\alpha^{\pi(i)}$ in commitments $\boldsymbol{c}_{\alpha^{\pi(i)}}$ (again indexed by i).

However the verifier still does not know what is contained in those commitments. We can temporarily call m_i to the message committed in $\boldsymbol{c}_{\pi(i)}$, and $\widehat{m}_i$ to the message committed in $\boldsymbol{c}_{\alpha^{\pi(i)}}$. The prover needs to convince the verifier that they are commitments to a permutation and to powers of α permuted by the same permutation.

To do so $\mathcal{V}$ chooses two more random polynomials $\beta,\gamma \xleftarrow{\$} S$. Using the Σ-proofs that allow us to prove linear relations $\mathcal{P}$ proves that he knows openings $m_i, \widehat{m}_i$ to commitments $\boldsymbol{c}_{\pi(i)}, \boldsymbol{c}_{\alpha^{\pi(i)}}$ that satisfy the following relation:

$$\prod_{i=1}^{N}\Big(\beta i + \alpha^i - \gamma\Big) = \prod_{i=1}^{N}(\beta m_i + \widehat{m}_i - \gamma) \tag{11}$$

We can consider the two expressions as polynomials in $R_q[G]$ evaluated in γ. The left hand side of the equation has been determined by the choices of the verifier, and for the right hand side, by the binding property of the commitment scheme we know that $m_i, \widehat{m}_i$ were determined before the choice for γ was made. The prover has shown that they are equal when evaluated in the specific $\gamma \in S$ chosen by the verifier, so we can apply the generalized Schwartz-Zippel lemma 4.3 and say that with overwhelming probability both polynomials defined by eq. (11) are equal in $R_q[\Gamma]$ and have the same roots. These roots may be in different order, defined by a permutation $\widetilde{\pi}$. For all $i \in 1, \dots, N$ we have:

$$\beta m_i + \widehat{m}_i = \beta\widetilde{\pi}(i) + \alpha^{\widetilde{\pi}(i)}, \quad \forall i \in 1, \dots, N$$

We can rewrite it as an equation on β:

$$\beta(m_i - \widetilde{\pi}(i)) = \alpha^{\widetilde{\pi}(i)} - \widehat{m}_i, \quad \forall i \in 1, \dots, N$$

The polynomials m_i and $\widehat{m}_i$ where fixed before β was chosen. But we still do not know if the permutation $\widetilde{\pi}$ was predetermined. However, looking at one i and fixed m_i and $\widehat{m}_i$ we can consider all possible $j \in 1, \dots, N$ and study $\beta(m_i - j) = \alpha^j - \widehat{m}_i$. If $(m_i - j) \neq 0$ then there exists at most one $\beta_j \in S$ that fulfills the equation with j (this was trivial in Bayer and Groth's proof, but in our case is again given by the condition of set S). The probability of choosing β equal to one of these (at most N) β_j is negligible. This implies that for each i there exists a j such that $m_i = j$ and therefore $\widehat{m}_i = \alpha^j$. With this reasoning for each i and the previous equations we finally get that, with overwhelming probability $m_i = \widetilde{\pi}(i)$ and $\widehat{m}_i = \alpha^{\widetilde{\pi}(i)}$.

This means that $\boldsymbol{c}_{\alpha^{\pi(i)}}$ are indeed commitments to α with exponents from 1 to N permuted in an order that was fixed by $\boldsymbol{c}_{\pi(i)}$ before α was chosen. Combining them with commitments to encryptions of 0 (this will be discussed later) we can build a ZK-proof involving the input and output of the *mix-net* node in the following polynomial relation:

$$\sum_{i=1}^{N} \alpha^i u^{(i)} = \sum_{i=1}^{N} \alpha^{\pi(i)} \left(u'^{(i)} - a r_{\mathsf{E}}'^{(i)} - e_{\mathsf{E},u}'^{(i)} \right) \tag{12}$$

Once again and for the last time we can see it as polynomials with coefficients in R_q and variable A that are equal when evaluated in α. If both polynomials were determined before α was picked up we can apply lemma 4.3 and conclude that with overwhelming probability they are equal as polynomials, and so:

$$u'^{(i)} = u^{\pi(i)} + a r_{\mathsf{E}}'^{(i)} + e_{\mathsf{E},u}'^{(i)}$$

And the same reasoning would apply to the $v'^{(i)}$.

$$v'^{(i)} = v^{\pi(i)} + b r_{\mathsf{E}}'^{(i)} + e_{\mathsf{E},v}'^{(i)}$$

The only remaining step, that could be executed before the election in an offline phase, is to prove that the ciphertexts that we add up are valid encryptions of 0. Random elements $r_{\mathsf{E}}'^{(i)}, e_{\mathsf{E},u}'^{(i)}, e_{\mathsf{E},v}'^{(i)}$ for $i \in 1, \dots, N$ are required to have coefficients belonging to $[-\delta + 1, \delta - 1]$. Those encryptions of 0 will be committed, and our real goal would be to prove in zero-knowledge that some polynomials are valid commitments to valid encryptions of 0. That is, that the pair of elements:

$$\left(\boldsymbol{c}_{u_0^{(i)}}, \boldsymbol{c}_{v_0^{(i)}} \right) = \left(\boldsymbol{a}_{\mathsf{C}} \left(a_{\mathsf{E}} r_{\mathsf{E}}'^{(i)} + e_{\mathsf{E},u}'^{(i)} \right) + \boldsymbol{b}_{\mathsf{C}} r_{\mathsf{C},u}^{(i)} + \boldsymbol{e}_{\mathsf{C},u}^{(i)}, \boldsymbol{a}_{\mathsf{C}} \left(b_{\mathsf{E}} r_{\mathsf{E}}'^{(i)} + e_{\mathsf{E},v}'^{(i)} \right) + \boldsymbol{b}_{\mathsf{C}} r_{\mathsf{C},v}^{(i)} + \boldsymbol{e}_{\mathsf{C},v}^{(i)} \right)$$

Is such that $r_{\mathsf{E}}^{\prime(i)}, e_{\mathsf{E},u}^{\prime(i)}, e_{\mathsf{E},v}^{\prime(i)}, \boldsymbol{e}_{\mathsf{C},u}^{(i)}, \boldsymbol{e}_{\mathsf{C},v}^{(i)}$ have small norm ($r_{\mathsf{C},u}^{(i)}, r_{\mathsf{C},v}^{(i)}$ can be any polynomial in R_q).
In order to prove this we are going to use the strategy proposed by del Pino and Lyubashebsky [dPL17].
We can write this as a linear function:

$$f\left(r_{\mathsf{E}}^{\prime(i)}, e_{\mathsf{E},u}^{\prime(i)}, e_{\mathsf{E},v}^{\prime(i)}, \boldsymbol{e}_{\mathsf{C},u}^{(i)}, \boldsymbol{e}_{\mathsf{C},v}^{(i)}, r_{\mathsf{C},u}^{(i)}, r_{\mathsf{C},v}^{(i)}\right) = \\ \left(\boldsymbol{a}_{\mathsf{C}}\left(a_{\mathsf{E}} r_{\mathsf{E}}^{\prime(i)} + e_{\mathsf{E},u}^{\prime(i)}\right) + \boldsymbol{b}_{\mathsf{C}} r_{\mathsf{C},u}^{(i)} + \boldsymbol{e}_{\mathsf{C},u}^{(i)}, \boldsymbol{a}_{\mathsf{C}}\left(b_{\mathsf{E}} r_{\mathsf{E}}^{\prime(i)} + e_{\mathsf{E},v}^{\prime(i)}\right) + \boldsymbol{b}_{\mathsf{C}} r_{\mathsf{C},v}^{(i)} + \boldsymbol{e}_{\mathsf{C},v}^{(i)}\right)$$

We want to prove that we know preimages of this function that satisfy some conditions, related to the norm of the preimages. Using the proposed method of [dPL17] we can take advantage of the fact that we need to prove knowledge preimages of the same function for all $i \in 1, \ldots, N$ and amortize the cost. Inside the proof we have to check different conditions for $(r_{\mathsf{E}}^{\prime(i)}, e_{\mathsf{E},u}^{\prime(i)}, e_{\mathsf{E},v}^{\prime(i)})$ and $(\boldsymbol{e}_{\mathsf{C},u}^{(i)}, \boldsymbol{e}_{\mathsf{C},u}^{(i)})$. This is not a problem, as we just need to treat them separately, computing different masking parameters and adjusting the probability of an abortion, and directly applying [dPL17] for each part.

4.2.1 Proof of a shuffle protocol

Combining all this parts in the proper order we can define our proposal for a lattice based proof of a shuffle, described in protocol 5.

Protocol 5: Mixnet protocol

$\mathcal{P}\left(u^{(i)}, v^{(i)}, u'^{(i)}, v'^{(i)}; \pi, r_{\mathsf{E}}'^{(i)}, e_{\mathsf{E},u}'^{(i)}, e_{\mathsf{E},v}'^{(i)}\right)$ $\qquad$ $\mathcal{V}\left(u^{(i)}, v^{(i)}, u'^{(i)}, v'^{(i)}\right)$

$\forall i \in 1, \dots, N, \quad \forall j \in 1, \dots, n, \quad \forall l \in 0, \dots, k-1$

$\boldsymbol{c}_{u_0^{(i)}} = \mathsf{Com}\left(a_{\mathsf{E}} r_{\mathsf{E}}'^{(i)} + e_{\mathsf{E},u}'^{(i)}\right)$

$\boldsymbol{c}_{v_0^{(i)}} = \mathsf{Com}\left(b_{\mathsf{E}} r_{\mathsf{E}}'^{(i)} + e_{\mathsf{E},v}'^{(i)}\right)$

$$\xrightarrow{\boldsymbol{c}_{u_0^{(i)}}, \boldsymbol{c}_{v_0^{(i)}}}$$

$$\text{ZK-proof}\left[\begin{array}{c|l} r_{\mathsf{E}}'^{(i)}, e_{\mathsf{E},u}'^{(i)}, e_{\mathsf{E},v}'^{(i)} \\ r_{\mathsf{C},u}^{(i)}, \boldsymbol{e}_{\mathsf{C},u}^{(i)}, r_{\mathsf{C},v}^{(i)}, \boldsymbol{e}_{\mathsf{C},v}^{(i)} & \begin{array}{c} \boldsymbol{c}_{u_0^{(i)}} = \boldsymbol{a}_{\mathsf{C}}\left(a_{\mathsf{E}} r_{\mathsf{E}}'^{(i)} + e_{\mathsf{E},u}'^{(i)}\right) + \boldsymbol{b}_{\mathsf{C}} r_{\mathsf{C},u}^{(i)} + \boldsymbol{e}_{\mathsf{C},u}^{(i)} \\ \boldsymbol{c}_{v_0^{(i)}} = \boldsymbol{a}_{\mathsf{C}}\left(b_{\mathsf{E}} r_{\mathsf{E}}'^{(i)} + e_{\mathsf{E},v}'^{(i)}\right) + \boldsymbol{b}_{\mathsf{C}} r_{\mathsf{C},v}^{(i)} + \boldsymbol{e}_{\mathsf{C},v}^{(i)} \\ \left\|r_{\mathsf{E}}'^{(i)}\right\|, \left\|e_j'^{(i)}\right\| \leq \tau\delta, \quad \left\|\boldsymbol{e}_{\mathsf{C},j}^{(i)}\right\| \leq \tau\delta' \end{array} \end{array}\right]$$

$\forall i \in 1, \dots, N$

$\boldsymbol{c}_{\pi(i)} = \mathsf{Com}(\pi(i))$

$$\xrightarrow{\boldsymbol{c}_{\pi(i)}}$$

$\alpha \xleftarrow{\$} S$

$$\xleftarrow{\alpha}$$

$\forall i \in 1, \dots, N$

$\boldsymbol{c}_{\alpha^{\pi(i)}} = \mathsf{Com}\left(\alpha^{\pi(i)}\right)$

$$\xrightarrow{\boldsymbol{c}_{\alpha^{\pi(i)}}}$$

$\beta, \gamma \xleftarrow{\$} S$

$$\xleftarrow{\beta, \gamma}$$

$$\text{ZK-proof}\left[\begin{array}{c|l} m_i, r_i, \boldsymbol{e}_{\mathsf{C},i}, f_i \\ \widehat{m}_i, \widehat{r}_i, \widehat{\boldsymbol{e}_{\mathsf{C}i}}, \widehat{f}_i & \begin{array}{c} \left(\prod_{i=1}^{N}\left(\beta i + \alpha^i - \gamma\right) = \prod_{i=1}^{N}\left(\beta m_i + \widehat{m}_i - \gamma\right)\right), \\ \bigwedge_{i=1}^{N}\left(\mathsf{Ver}(\boldsymbol{c}_{\pi(i)}; m_i, r_i, \boldsymbol{e}_{\mathsf{C},i}, f_i) = \text{accept}\right), \\ \bigwedge_{i=1}^{N}\left(\mathsf{Ver}(\boldsymbol{c}_{\alpha^{\pi(i)}}; \widehat{m}_i, \widehat{r}_i, \widehat{\boldsymbol{e}_{\mathsf{C}i}}, \widehat{f}_i) = \text{accept}\right) \end{array} \end{array}\right]$$

$$\text{ZK-proof}\left[\begin{array}{c|l} y \in \left\{\begin{smallmatrix}\alpha^{\pi(i)} \\ u_0^{(i)}\end{smallmatrix}\right\}_i & \sum_{i=1}^{N}\alpha^i u^{(i)} = \sum_{i=1}^{N} m_{\alpha^{\pi(i)}}\left(u'^{(i)} - m_{u_0^{(i)}}\right) \\ r_y & \bigwedge_y\left(\mathsf{Ver}(\boldsymbol{c}_y; m_y, r_y, \boldsymbol{e}_{\mathsf{C},y}, f_y) = \text{accept}\right) \\ \boldsymbol{e}_{\mathsf{C},y} & \\ f_y & \end{array}\right]$$

$$\text{ZK-proof}\left[\begin{array}{c|l} y \in \left\{\begin{smallmatrix}\alpha^{\pi(i)} \\ v_0^{(i)}\end{smallmatrix}\right\}_i & \sum_{i=1}^{N}\alpha^i v^{(i)} = \sum_{i=1}^{N} m_{\alpha^{\pi(i)}}\left(v'^{(i)} - m_{v_0^{(i)}}\right) \\ r_y & \bigwedge_y\left(\mathsf{Ver}(\boldsymbol{c}_y; m_y, r_y, \boldsymbol{e}_{\mathsf{C},y}, f_y) = \text{accept}\right) \\ \boldsymbol{e}_{\mathsf{C},y} & \\ f_y & \end{array}\right]$$

outputs accept if all ZK-proofs are correct

4.2.2 Completeness, Zero-Knowledge and Soundness

If the prover $\mathcal{P}$ choses all re-encryption parameters from the appropriate distributions χ_σ conditioned to have norm smaller than δ, correctly builds the commitments to the encryptions of 0 and follows the small secrets proof the answer will be accepted. This is also the case for the proof of the committed permuted powers of α, as products $\prod_{i=1}^{N}(\beta i + \alpha^i - \gamma)$ and $\prod_{i=1}^{N}(\beta m_i + \widehat{m}_i - \gamma)$ are exactly equal, just in permuted order. Finally the two last ZK-proofs are accepted as the output is exactly a permutation and re-encryption of the input, and we have built a polynomial subtracting the re-encryptions and inverting the permutation, therefore it is also accepted. To summarize, the protocol is accepted as all the ZK-proofs involved are accepted if an honest prover follows the protocols, so it is complete.

All information published consists of computationally hiding commitments and ZK-proof conversations that can be simulated, which implies that the whole protocol is special honest-verifier zero-knowledge.

Soundness comes from the generalized Schwartz-Zippel lemma, the binding property of Benhamouda *et al.* commitments and the soundness of each of the ZK-proofs.

We start with the first ZK-proof.

$$\text{ZK-proof}\left[\begin{array}{c} r_{\mathsf{E}}'^{(i)}, e_{\mathsf{E},u}'^{(i)}, e_{\mathsf{E},v}'^{(i)} \\ r_{\mathsf{C},u}^{(i)}, \boldsymbol{e}_{\mathsf{C},u}^{(i)}, r_{\mathsf{C},v}^{(i)}, \boldsymbol{e}_{\mathsf{C},v}^{(i)} \end{array} \middle| \begin{array}{c} \boldsymbol{c}_{u_0^{(i)}} = \boldsymbol{a}_{\mathsf{C}}\left(a_{\mathsf{E}} r_{\mathsf{E}}'^{(i)} + e_{\mathsf{E},u}'^{(i)}\right) + \boldsymbol{b}_{\mathsf{C}} r_{\mathsf{C},u}^{(i)} + \boldsymbol{e}_{\mathsf{C},u}^{(i)} \\ \boldsymbol{c}_{v_0^{(i)}} = \boldsymbol{a}_{\mathsf{C}}\left(b_{\mathsf{E}} r_{\mathsf{E}}'^{(i)} + e_{\mathsf{E},v}'^{(i)}\right) + \boldsymbol{b}_{\mathsf{C}} r_{\mathsf{C},v}^{(i)} + \boldsymbol{e}_{\mathsf{C},v}^{(i)} \\ \left\|r_{\mathsf{E}}'^{(i)}\right\|, \left\|e_j'^{(i)}\right\| \leq \tau\delta, \quad \left\|\boldsymbol{e}_{\mathsf{C},j}^{(i)}\right\| \leq \tau\delta' \end{array}\right]$$

If δ' is such that $\tau\delta' \leq \left\lfloor \frac{n^{4/3}}{2} \right\rfloor$ the extractor of this zero-knowledge proof given by Del Pino and Lyubashebsky provides us with valid openings of $\boldsymbol{c}_{u_0^{(i)}}$ and $\boldsymbol{c}_{v_0^{(i)}}$ to a valid encryption of 0. Then, we analyze the next zero-knowledge proof.

$$\text{ZK-proof}\left[\begin{array}{c} m_i, r_i, \boldsymbol{e}_{\mathsf{C},i}, f_i \\ \widehat{m}_i, \widehat{r}_i, \widehat{\boldsymbol{e}_{\mathsf{C}i}}, \widehat{f}_i \end{array} \middle| \begin{array}{c} \left(\prod_{i=1}^{N}\left(\beta i + \alpha^i - \gamma\right) = \prod_{i=1}^{N}\left(\beta m_i + \widehat{m}_i - \gamma\right)\right), \\ \bigwedge_{i=1}^{N}\left(\mathsf{Ver}(\boldsymbol{c}_{\pi(i)}; m_i, r_i, \boldsymbol{e}_{\mathsf{C},i}, f_i) = \text{accept}\right), \\ \bigwedge_{i=1}^{N}\left(\mathsf{Ver}(\boldsymbol{c}_{\alpha^{\pi(i)}}; \widehat{m}_i, \widehat{r}_i, \widehat{\boldsymbol{e}_{\mathsf{C}i}}, \widehat{f}_i) = \text{accept}\right) \end{array}\right]$$

This time, using the extractor of Benhamouda *et al.* we obtain valid openings for $\boldsymbol{c}_{\pi(i)}$ and $\boldsymbol{c}_{\alpha^{\pi(i)}}$ that satisfy the equation $\prod_{i=1}^{N}(\beta i + \alpha^i - \gamma) = \prod_{i=1}^{N}(\beta m_i + \widehat{m}_i - \gamma)$. The order in which all polynomials have been determined, generalized Schwartz-Zippel and the previously discussed argument guarantees that, with overwhelming probability those extracted messages are permuted integers from 1 to N and powers of α in the same order. Finally we have the last two zero-knowledge proofs.

$$\text{ZK-proof}\left[\begin{array}{c} y \in \left\{\begin{smallmatrix}\alpha^{\pi(i)} \\ u_0^{(i)}\end{smallmatrix}\right\}_i \\ r_y \\ \boldsymbol{e}_{\mathsf{C},y} \\ f_y \end{array} \middle| \begin{array}{c} \sum_{i=1}^{N}\alpha^i u^{(i)} = \sum_{i=1}^{N} m_{\alpha^{\pi(i)}}\left(u'^{(i)} - m_{u_0^{(i)}}\right) \\ \bigwedge_{y}\left(\mathsf{Ver}(\boldsymbol{c}_y; m_y, r_y, \boldsymbol{e}_{\mathsf{C},y}, f_y) = \text{accept}\right) \end{array}\right]$$

$$\text{ZK-proof}\left[\begin{array}{c|c} y \in \left\{\begin{smallmatrix}\alpha^{\pi(i)}\\ v_0^{(i)}\end{smallmatrix}\right\}_i & \sum_{i=1}^{N}\alpha^i v^{(i)} = \sum_{i=1}^{N} m_{\alpha^{\pi(i)}}\left(v'^{(i)} - m_{v_0^{(i)}}\right) \\ r_y & \bigwedge_y\left(\text{Ver}(\boldsymbol{c}_y; m_y, r_y, \boldsymbol{e}_{\text{C},y}, f_y) = \text{accept}\right) \\ \boldsymbol{e}_{\text{C},y} & \\ f_y & \end{array}\right]$$

Using the extractor of these proofs we obtain openings of $\boldsymbol{c}_{\pi(i)}, \boldsymbol{c}_{\alpha^{\pi(i)}}, \boldsymbol{c}_{u_0^{(i)}}, \boldsymbol{c}_{v_0^{(i)}}$. Given that the commitment scheme is binding we know from previous proofs that those openings are $\pi(i), \alpha^{\pi(i)}, u_0^{(i)}, v_0^{(i)}$. Then, the relations hold by the messages committed that were written in terms of m_y are exactly:

$$\sum_{i=1}^{N}\alpha^i u^{(i)} = \sum_{i=1}^{N}\alpha^{\pi(i)}\left(u'^{(i)} - u_0^{(i)}\right)$$
$$\sum_{i=1}^{N}\alpha^i v^{(i)} = \sum_{i=1}^{N}\alpha^{\pi(i)}\left(v'^{(i)} - v_0^{(i)}\right)$$

Applying for the last time the generalized Schwartz-Zippel lemma we can ensure with overwhelming probability that:

$$\forall i \in 1, \ldots, N$$
$$u^{(i)} = u'^{\pi^{-1}(i)} - u_0^{\pi^{-1}(i)}$$
$$v^{(i)} = v'^{\pi^{-1}(i)} - v_0^{\pi^{-1}(i)}$$

And it implies that the mix-net node has performed a correct shuffle on the input votes.

5. Conclusions

While quantum computers are a big threat to classical cryptography, post-quantum cryptographic primitives are experiencing a boost of publications in the recent years. Post-quantum alternatives may soon be mature enough to replace them, and lattice based cryptography is the most promising alternative for public key encryption schemes. It is of paramount importance to start basing our communications on post-quantum assumptions, in order to prevent future attacks on todays data and to have solid cryptographic constructions ready.

In this thesis we have presented two protocols necessary to build an electronic voting scheme. The first one is a direct application of existing lattice based constructions, and allows a voter to check that his vote has been casted correctly. Then a proof of a shuffle is derived from existing protocols, in this case not being a direct translation as many modifications have been needed to adapt it to lattice based requirements.

In order to define a complete voting scheme they have to be combined with other protocols for different requirements, for example a key exchange mechanism for threshold decryption, a way of signing the votes that allows the voting server to recognize that it has been casted by an authorized voter, a proof of validity that ensures that a casted vote is a valid vote and a mechanism that prevents other voters from casting the same vote (or a modification of the vote) of another user. These protocols (that might be combined) have to be specifically defined, and are left for future work.

Our proposal for a proof of a shuffle has size linear in the number of votes. A real implementation taking advantage of all possible optimizations (FFT and efficient discrete gauss sampling) would be needed to see how well does it compare with non post-quantum proposals. Another future line of work would be to improve the efficiency of some of the constructions. The amortized proof of knowledge of small secrets seems to be the one with more possible improvements, as many papers with incremental improvements have been proposed recently regarding this subject. On the other hand, the proof of Ling *et al.* derives from a code based proof by Stern, and several improvements have been made recently on the code framework [AGS11], that are worth studying in case those improvements have a translation on the lattice framework.

While it has been developed for several years post-quantum cryptography is still a very recent area, for which there is a lot of work to do.

References

[Abe98] Masayuki Abe. Universally verifiable mix-net with verification work independent of the number of mix-servers. EUROCRYPT 1998, pages 437–447. Springer, 1998.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange-a new hope. In *USENIX Security Symposium*, pages 327–343, 2016.

[AGS11] Carlos Aguilar, Philippe Gaborit, and Julien Schrek. A new zero-knowledge code based identification scheme with reduced communication. In *Information Theory Workshop (ITW), 2011 IEEE*, pages 648–652. IEEE, 2011.

[AH01] Masayuki Abe and Fumitaka Hoshino. Remarks on mix-network based on permutation networks. In *Public Key Cryptography*, PKC'01, pages 317–324. Springer, 2001.

[Ajt98] Miklós Ajtai. The shortest vector problem in l 2 is np-hard for randomized reductions. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 10–19. ACM, 1998.

[ARU14] Andris Ambainis, Ansis Rosmanis, and Dominique Unruh. Quantum attacks on classical proof systems: The hardness of quantum rewinding. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 474–483. IEEE, 2014.

[BG12] Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In *Advances in Cryptology*, EUROCRYPT 2012, pages 263–280. Springer, 2012.

[BGM93] Ian F. Blake, Shuhong Gao, and Ronald C. Mullin. Explicit factorization of $x^{2^k}+1$ over $\mathbb{F}_p$ with prime $p \equiv 3 \mod 4$. *Applicable Algebra in Engineering, Communication and Computing*, 4(2):89–94, 1993.

[BKLP15] Fabrice Benhamouda, Stephan Krenn, Vadim Lyubashevsky, and Krzysztof Pietrzak. Efficient zero-knowledge proofs for commitments from learning with errors over rings. In *European Symposium on Research in Computer Security*, ESORICS 2015, pages 305–325. Springer, 2015.

[BL17] Carsten Baum and Vadim Lyubashevsky. Simple amortized proofs of shortness for linear relations over polynomial rings. Technical report, IACR Cryptology ePrint Archive, 2017: 759, 2017.

[Bra16] Matt Braithwaite. Experimenting with post-quantum cryptography. `https://security.googleblog.com`, 2016.

[CDXY17] Ronald Cramer, Ivan Damgård, Chaoping Xing, and Chen Yuan. Amortized complexity of zero-knowledge proofs revisited: Achieving linear soundness slack. In *Advances in Cryptology*, EUROCRYPT 2017, pages 479–500. Springer, 2017.

[Cha81] David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[CMM17] Nuria Costa, Ramiro Martínez, and Paz Morillo. Proof of a shuffle for lattice-based cryptography. In *Nordic Conference on Secure IT Systems*, NORDSEC 2017, pages 280–296. Springer, 2017.

[CoE17] Committee of Ministers Council of Europe. Recommendation cm/rec(2017)5 of the committee of ministers to member states on standards for e-voting, 2017.

[dPL17] Rafael del Pino and Vadim Lyubashevsky. Amortization with fewer equations for proving knowledge of small secrets. In *Advances in Cryptology*, CRYPTO 2017, pages 365–394. Springer, 2017.

[Dwo15] Morris J Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions. *Federal Inf. Process. Stds.(NIST FIPS)-202*, 2015.

[FP85] Ulrich Fincke and Michael Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of computation*, 44(170):463–471, 1985.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the Theory and Application of Cryptographic Techniques*, CRYPTO' 86, pages 186–194. Springer, 1986.

[FS01] Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In *Proc. of the 21st Annual Int. Cryptology Conf. on Advances in Cryptology*, volume 1 of *CRYPTO '01*, pages 368–387. Springer, 2001.

[Fur05] Jun Furukawa. Efficient and verifiable shuffling and shuffle-decryption. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 88(1):172–188, 2005.

[GC16] Sandra Guasch Castelló. *Individual verifiability in electronic voting*. PhD thesis, Universitat Politècnica de Catalunya, 2016.

[GGH11] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Collision-free hashing from lattice problems. *Studies in complexity and cryptography*, 6650:30–39, 2011.

[GI08] Jens Groth and Yuval Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In *Advances in Cryptology*, EUROCRYPT 2008, pages 379–396. Springer, 2008.

[GN08] Nicolas Gama and Phong Q Nguyen. Finding short lattice vectors within mordell's inequality. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 207–216. ACM, 2008.

[GNR] Nicolas Gama, Phong Q Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In *Advances in Cryptology*, EUROCRYPT 2010.

[Gro03] Jens Groth. A verifiable secret shuffle of homomorphic encryptions. In *Proc. of the 6th Int. Workshop on Theory and Practice in Public Key Cryptography*, volume 2567, pages 145–160. Springer, 2003.

[Kan83] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 193–206. ACM, 1983.

[KR97] Hugo Krawczyk and Tal Rabin. Chameleon hashing and signatures. 1997.

[LLL82] Arjen Klaas Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[LLM+16] Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for matrix-vector relations and lattice-based group encryption. In *Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II 22*, pages 101–131. Springer, 2016.

[LNSW13] San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. Improved zero-knowledge proofs of knowledge for the isis problem, and applications. In *PKC 2013: 16th Int. Conf. on Practice and Theory in Public-Key Cryptography*, volume 7778, pages 107–124. Springer, 2013.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, 60(6):43, 2013.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In *Advances in Cryptology*, EUROCRYPT 2012, pages 738–755. Springer, 2012.

[Mas99] Abe Masayuki. Mix-networks on permutation networks. In *Proc. of the Int. Conf. on the Theory and Applications of Cryptology and Information Security*, ASIACRYPT 1999, page 258. Springer, 1999.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology – EUROCRYPT 2012*, volume 7237, pages 700–718. Springer, 2012.

[Nef01] C Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 116–125. ACM, 2001.

[Nef03] C. Andrew Neff. Verifiable mixing (shuffling) of elgamal pairs. *VHTi Technical Document, http://www.votehere.net/vhti/documentation/egshuf.pdf, VoteHere, Inc*, 2003.

[NV08] Phong Q Nguyen and Thomas Vidick. Sieve algorithms for the shortest vector problem are practical. *Journal of Mathematical Cryptology*, 2(2):181–207, 2008.

[PCGK17] Jordi Puiggalí, Jordi Cucurull, Sandra Guasch, and Robert Krimmer. Verifiability experiences in government online voting systems. In *International Joint Conference on Electronic Voting*, pages 248–263. Springer, 2017.

[Ped91] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology*, volume 91 of *CRYPTO 1991*, pages 129–140. Springer, 1991.

[PIK94] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *Advances in Cryptology -— EUROCRYPT'93*, pages 248–259. Springer, 1994.

[SE94] Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1-3):181–199, 1994.

[SG16] Eduard Sanou Gozalo. Post-quantum cryptography: lattice-based encryption. Master's thesis, Universitat Politècnica de Catalunya, 2016.

[Sho99] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme. In *Advances in Cryptology—EUROCRYPT'95*, pages 393–403. Springer, 1995.

[SRB14] Kunwar Singh, C Pandu Rangan, and AK Banerjee. Lattice based universal re-encryption for mixnet. *J. Internet Serv. Inf. Secur.*, 4(1):1–11, 2014.

[SRB15] Kunwar Singh, C Pandu Rangan, and AK Banerjee. Lattice based mix network for location privacy in mobile system. *Mobile Information Systems*, 2015, 2015.

[Ste96] Jacques Stern. A new paradigm for public key identification. *IEEE Transactions on Information Theory*, 42(6):1757–1768, 1996.

[TW10] Björn Terelius and Douglas Wikström. Proofs of restricted shuffles. In *AFRICACRYPT '10: Third Int. Conf. on Cryptology in Africa, Stellenbosch, South Africa*, volume 10, pages 100–113. Springer, 2010.

[Unr15] Dominique Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In *Advances in Cryptology*, EUROCRYPT 2015, pages 755–784, 2015.

[Wik04] Douglas Wikstrom. A universally composable mix-net. In *First Theory of Cryptography Conf., TCC 2004*, volume 2951, pages 317–335. Springer, 2004.

[Wik05] Douglas Wikström. A sender verifiable mix-net and a new proof of a shuffle. In *International Conference on the Theory and Application of Cryptology and Information Security*, ASIACRYPT'05, pages 273–292. Springer, 2005.

[Wik09] Douglas Wikström. A commitment-consistent proof of a shuffle. In *Proc. of the 14th Australasian Conf. on Information Security and Privacy*, volume 9 of *ACISP '09*, pages 407–421. Springer, 2009.

[WLTB11] Xiaoyun Wang, Mingjie Liu, Chengliang Tian, and Jingguo Bi. Improved nguyen-vidick heuristic sieve algorithm for shortest vector problem. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 1–9. ACM, 2011.

[XXW13] Xiang Xie, Rui Xue, and Minqian Wang. Zero knowledge proofs from ring-lwe. In *International Conference on Cryptology and Network Security*, pages 57–73. Springer, 2013.